ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение
высшего профессионального образования
УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет математики и информационных технологий

Ю. В. Цыганова

Языки и методы программирования

Лабораторный практикум
Раздел: Методы трансляции языков программирования

Учебно-методическое пособие

Ульяновск – 2011

УДК 004.43 (075.8)
ББК 32.973.2- 018 я73
Ц 94

Печатается по решению Ученого Совета
факультета математики и информационных технологий
Ульяновского государственного университета

Рецензент:

Зав. кафедрой информатики УлГПУ им. И. Н. Ульянова
доцент, кандидат педагогических наук Е. В. Беляева

Ц 94 Цыганова Ю.В.
Языки и методы программирования. (Лабораторный практикум. Раздел: Методы трансляции языков программирования): учебно-методическое пособие / Ю.В. Цыганова. – Ульяновск: УлГУ, 2011. – 39 с.

Данное учебное пособие представляет собой лабораторный практикум по дисциплине «Языки и методы программирования», раздел «Методы трансляции языков программирования».

Пособие состоит из введения, описания индивидуального программного проекта, четырех лабораторных работ и списка литературы, оно входит в комплект методических материалов для обучения в третьем семестре на факультете математики и информационных технологий Ульяновского государственного университета. Предназначено для студентов специальностей «Прикладная математика и информатика», «Компьютерная безопасность», «Информационные системы». Пособие содержит 65 вариантов заданий.

УДК 004.43 (075.8)
ББК 32.973.2- 018 я73

СОДЕРЖАНИЕ

Введение

Учебная дисциплина «Языки и методы программирования» читается на первом и втором курсах факультета математики и информационных технологий Ульяновского государственного университета для специальностей «Прикладная математика и информатика» и «Компьютерная безопасность», «Информационные системы». В третьем семестре в рамках данного курса изучается раздел: «Методы трансляции языков программирования». Данный раздел поддержан лабораторным практикумом.

На лабораторных занятиях студентам предлагается выполнить лабораторный проект согласно индивидуальному варианту. Лабораторный проект представляет собой набор логически связанных между собой лабораторных работ, каждая из которых представляет собой реализацию отдельной фазы трансляции программы с конкретного языка программирования (представленного набором синтаксических конструкций). Таким образом, весь лабораторный проект – это программная реализация полного цикла транслятора языка программирования.

Задания максимально приближены к решению реальной задачи разработки трансляторов для языков программирования высокого уровня, что позволяет студентам проверить, изучить и применить на практике полученные теоретические знания.

Данное пособие содержит 65 вариантов заданий на лабораторный проект, что позволяет каждому студенту на курсе работать по индивидуальному заданию. Основой лабораторного практикума явился учебно-методический материал, разработанный доктором технических наук А. И. Легаловым[1]. В

[1] www.softcraft.ru

данном учебном пособии все материалы используются в существенно переработанном виде.

1. Общие требования к выполнению лабораторного проекта

Лабораторный проект выполняется на языке C++ в виде пакета программ, каждая из которых является реализацией отдельной фазы транслятора. Выходные данные каждой лабораторной работы являются входными данными для следующей лабораторной работы. Диаграммы Вирта изображаются в рукописном виде в тетради, тестовые задачи оформляются в рукописном виде в тетради и в электронном виде как текстовые файлы.

Критерии оценки лабораторного практикума:

Лабораторная работа №1 – 30 баллов;

Лабораторная работа №2 – 20 баллов;

Лабораторная работа №3 – 25 баллов;

Лабораторная работа №4 – 20 баллов.

Таким образом, за полностью выполненный лабораторный проект студент получает 95 баллов, которые учитываются при выставлении базовой оценки за экзамен.

2. Правила составления индивидуального задания на лабораторный проект

На первом лабораторном занятии студент получает от преподавателя номер варианта лабораторного проекта. Согласно этому номеру студент должен составить и принести на проверку преподавателю индивидуальное

задание на лабораторный проект. Индивидуальное задание оформляется в виде файла в формате *.doc и затем распечатывается на принтере. Индивидуальное задание обязательно должно содержать:

1. Титульный лист (см. стр. 12).
2. Набор синтаксических конструкций согласно таблице на стр. 36. В таблице необходимо найти строку с номером своего варианта, в шапке таблице указаны номера разделов, в строке – номера пунктов, соответствующие номеру варианта.
3. Набор тестовых задач. Номер варианта тестовых задач содержится в последнем столбце таблице на стр. 35-37, варианты тестовых задач – на стр. 32-34.

3. Лабораторная работа № 1

Описание синтаксиса языка программирования с использованием диаграмм Вирта

Цель работы

Изучение основ теории языков и формальных грамматик, метаязыков, методов описания пользовательского синтаксиса. Использование Диаграмм Вирта для описания синтаксиса языка программирования.

Порядок выполнения

1. Ознакомиться с описанием лабораторной работы и необходимым теоретическим материалом.
2. В файле **`МТ_Варианты.doc`** определить вариант задания в соответствии с полученным от преподавателя номером.
3. Перевести формальное описание разрабатываемого языка программирования из РБНФ в диаграммы Вирта.

4. Написать четыре содержательных примера программ (в соответствии со своим вариантом), раскрывающих особенности использования конструкций данных разрабатываемого языка, отразив в этих примерах все его функциональные возможности.
5. Представить отчет о проделанной работе.

Содержание отчета

1. Синтаксис языка в соответствии со своим вариантом, выполненный с использованием РБНФ. Сдается на проверку в виде электронного документа.
2. Пользовательское описание синтаксиса разрабатываемого языка, построенное с использованием диаграмм Вирта. Должно быть представлено в рукописном виде в тетради. Требуется аккуратное оформление в соответствии с правилами изображения диаграмм Вирта.
3. Четыре содержательных примера программ для работы с различными типами данных и с использованием различных операторов. Они должны быть представлены в рукописном виде в тетради и в виде отдельных текстовых файлов, что в дальнейшем позволяет использовать их при тестировании транслятора.

4. Лабораторная работа № 2

Разработка лексического анализатора заданного языка программирования

Цель работы

Изучение основ теории языков и формальных грамматик, методов построения лексических анализаторов (или сканеров).

Порядок выполнения

1. Ознакомиться с описанием лабораторной работы и необходимым теоретическим материалом.
2. Написать программу, реализующую сканер, в соответствии со своим вариантом. В состав сканера обязательно должен входить транслитератор.
3. Вход сканера: текстовый файл с программой на Вашем языке программирования, выход: файл с отчетом, содержащий описание каждой прочитанной лексемы, либо сообщения об ошибках.
4. Проверить работу сканера на разработанных тестовых примерах.
5. Представить отчет о проделанной работе.

Содержание отчета

Исходный текст программы сканера.

5. Лабораторная работа № 3

Разработка синтаксического анализатора для заданного языка программирования

Цель работы

Изучение основ теории языков и формальных грамматик, методов построения синтаксических анализаторов.

Порядок выполнения

1. Ознакомиться с описанием лабораторной работы и необходимым теоретическим материалом.
2. Написать программу, реализующую синтаксический анализатор (или распознаватель), в соответствии со своим вариантом. Алгоритм работы распознавателя по Вашему выбору (например, метод рекурсивного спуска).
3. Вход синтаксического анализатора: текстовый файл с программой на Вашем языке программирования и выходной файл сканера, выход: файл с отчетом, содержащий таблицу переменных, либо сообщения об ошибках.
4. Проверить работу синтаксического анализатора на разработанных тестовых примерах.
5. Представить отчет о проделанной работе.

Содержание отчета

Исходный текст программы распознавателя.

Примерный перечень сообщений об ошибках:

1. Равно вместо присваивания
2. Нет числа после знака равно
3. Нет знака операции после идентификатора
4. Нет идентификатора после **`var`**
5. Пропущена запятая или точка с запятой
6. Нет оператора
7. Пропущена точка с запятой между операторами
8. Неописанный идентификатор
9. Требуется **`then`**
10. Требуется точка с запятой или **end**

11.Неверный символ после оператора

12.Требуется сравнение

13.Отсутствует правая скобка

14.Неверный символ после множителя

15.Неверный символ в начале выражения

16.Слишком большое число

17. и т. д.

6. Лабораторная работа № 4

Разработка интерпретатора для заданного языка программирования.

Цель работы

Изучение основ теории языков и формальных грамматик, методов построения интерпретаторов.

Порядок выполнения

1. Ознакомиться с описанием лабораторной работы и необходимым теоретическим материалом.
2. Написать программу, реализующую интерпретатор, в соответствии со своим вариантом. Алгоритм работы интерпретатора разработать самостоятельно.
3. Вход интерпретатора: текстовый файл с программой на Вашем языке программирования и выходной файл распознавателя, выход: результат исполнения программы, занесенный в текстовый файл. Файл с отчетом должен содержать имя каждой программной переменной и ее значение на каждом шаге исполнения программы. Интерпретатор также должен

иметь окно вывода (output), в котором при исполнении программы нужно отобразить все данные, полученные с помощью операторов ввода и вывода. Кроме этого, интерпретатор должен выявлять логические ошибки, или ошибки вычислений (например, деление на ноль, выход за пределы массива, переполнение и т. д.) и выводить о них сообщение в окно output.

4. Проверить работу интерпретатора на разработанных тестовых примерах.
5. Представить отчет о проделанной работе.

Содержание отчета

Исходный текст программы интерпретатора.

7. Варианты индивидуального задания на лабораторный проект

Титульный лист:

Ульяновский государственный университет

Индивидуальное задание

на лабораторный проект

по курсу “Языки и методы программирования”

Студента_____ Группы _____

Вариант_____

Ульяновск 2011

Варианты описания языка программирования

I. Правила, используемые в лексическом анализаторе

I.1. Тип лексического анализатора (сканера)

Существуют два основных метода лексического анализа:

I.1.1 прямой

I.1.2 непрямой.

Они рассматриваются в теме: "Лексический анализ". Этот тип лексического анализатора будет использоваться в лабораторной работе для преобразования пользовательского синтаксиса к виду, определяемому заданным значением.

I.2. Наборы ключевых слов и разделителей

Разрабатываемые языки программирования отличаются используемыми ключевыми словами и разделителями.

Варианты стиля ключевых слов и разделителей:

I.2.1 **Паскаль.**

Таблица

Назначение специфических ключевых слов и разделителей

Назначение	**Обозначение конструкции языка**	**Обозначение лексемы**
Сложение	+	ADD
Вычитание	-	MIN
Умножение	*	MULT
Деление	/	DIV
Остаток по модулю	mod	MOD
Равно	=	EQ
Не равно	<>	NE
Меньше	<	LT
Больше	>	GT
Меньше или равно	<=	LE
Больше или равно	>=	GE
Присваивание	:=	ASG
Начало комментария	{	COMMENT
Конец комментария	}	

Начало составного	begin	BST
Конец составного	end	EST
Разделитель операторов	;	EOP

I.2.2. **Си**

Таблица

Назначение специфических ключевых слов и разделителей

Назначение	**Обозначение конструкции языка**	**Обозначение лексемы**
Сложение	+	ADD
Вычитание	-	MIN
Умножение	*	MULT
Деление	/	DIV
Остаток по модулю	%	MOD
Равно	==	EQ
Не равно	!=	NE
Меньше	<	LT
Больше	>	GT
Меньше или равно	<=	LE
Больше или равно	>=	GE

Присваивание	=	ASG
Начало комментария	/*	COMMENT
Конец комментария	*/	
Начало составного	{	BST
Конец составного	}	EST
Разделитель операторов	;	EOP

I.2.3 **Бейсик.**

Таблица

Назначение специфических ключевых слов и разделителей

Назначение	**Обозначение конструкции языка**	**Обозначение лексемы**
Сложение	+	ADD
Вычитание	-	MIN
Умножение	*	MULT
Деление	/	DIV
Остаток по модулю	mod	MOD
Равно	=	EQ
Не равно	<>	NE
Меньше	<	LT

Больше	>	GT
Меньше или равно	<=	LE
Больше или равно	>=	GE
Присваивание	=	ASG
Начало комментария	REM	COMMENT
Конец комментария	конец строки	
Разделитель операторов	;	EOP

I.2.4. Лисп

Таблица

Назначение специфических ключевых слов и разделителей

Назначение	**Обозначение конструкции языка**	**Обозначение лексемы**
Сложение	PLUS	ADD
Вычитание	MINUS	MIN
Умножение	MULT	MULT
Деление	DIV	DIV
Остаток по модулю	MOD	MOD
Равно	EQ	EQ
Не равно	NE	NE
Меньше	LT	LT

Больше	GT	GT
Меньше или равно	LE	LE
Больше или равно	GE	GE
Присваивание	LET	ASG
Начало комментария	//	COMMENT
Конец комментария	Признак конца строки	
Начало составного	DO	BST
Конец составного	END	EST
Разделитель операторов	;	EOP

I.2.5. Фортран

Таблица

Назначение специфических ключевых слов и разделителей

Назначение	**Обозначение конструкции языка**	**Обозначение лексемы**
Сложение	+	ADD
Вычитание	-	MIN
Умножение	*	MULT
Деление	/	DIV
Остаток по модулю	DMOD	MOD
Равно	.EQ.	EQ

Не равно	.NE.	NE
Меньше	.LT.	LT
Больше	.GT.	GT
Меньше или равно	.LE.	LE
Больше или равно	.GE.	GE
Присваивание	=	ASG
Начало комментария	**	COMMENT
Конец комментария	конец строки	
Разделитель операторов	;	EOP

I.3. Идентичность прописных и строчных букв (верхнего и нижнего регистров)

Существуют языки программирования, в которых эквивалентные по значению ключевые слова, идентификаторы, метки и другие имена можно задавать с любым сочетанием прописных и строчных букв, так как регистр символа игнорируется. В качестве примера можно назвать Бейсик и Паскаль. В этом случае в блоке лексического анализа имеется специальный модуль, преобразующий поступающие буквы к одному регистру (верхний или нижний регистр выбираются произвольно или с учетом соглашений, принятых в операционной системе при работе с внешними устройствами). В других языках, например Си, верхний и нижний регистр различаются. В этом случае регистр ключевых слов отдается на усмотрение разработчика.

Варианты:

I.3.1. **прописные и строчные буквы различаются.**

I.3.2. **прописные и строчные буквы не различаются.**

I.4. Правила, определяющие идентификатор, букву и цифру

Варианты: I.4.1.

```
$идентификатор=буква{буква|цифра}.

$буква="A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"I"|"J"|"K"|

"L"|"M"|"N"|"O"|"P"|"Q"|"R"|"S"|"T"|"U"|"V"|"W"|"X"|

"Y"|"Z"|"a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|

"l"|"m"|"n"|"o"|"p"|"q"|"r"|"s"|"t"|"u"|"v"|"w"|"x"|

"y"|"z".

$цифра="0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".
```

I.5. Организация метки

Метка состоит из ее имени и двоеточия, являющегося ограничителем данной конструкции. Имя метки может быть буквенно-цифровой строкой или целым десятичным числом. Предполагается, что во втором случае имя метки преобразуется в натуральное число в допустимом диапазоне. Поэтому, для одной и той же метки ее представление может отличаться по числу незначащих нулей. Например, метка "1995:" эквивалентна "00001995:".

Варианты:

I.5.1.

Метка - буквенно-цифровая строка:

```
$метка=имя_метки":".
```

```
$имя_метки=буква{буква|цифра}.
```

I.5.2.

Метка - строка десятичных цифр:

```
$метка=имя_метки":".

$имя_метки={/цифра/}.
```

I.6. Организация целых чисел

Предлагается три варианта, отличающихся друг от друга способом представления двоичных, восьмеричных, десятичных и шестнадцатеричных чисел.

Варианты:

I.6.1. (Си)

```
$двоичное="0"("B"|"b"){/"0"|"1"/}.

$восьмеричное=

"0"("C"|"c"){/"0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"/}.

$десятичное={/цифра/}.

$шестнадцатеричное=

"0"("X"|"x"){/цифра|"A"|"B"|"C"|"D"|"E"|"F"|"a"|"b"|"c"|

"d"|"e"|"f"/}.
```

I.6.2. (Ассемблер)

```
$двоичное={/"0"|"1"/}("B"|"b").

$восьмеричное=

{/"0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"/}("C"|"c").

$десятичное={/цифра/}["D"|"d"].
```

```
$шестнадцатеричное=
цифра{цифра|"A"|"B"|"C"|"D"|"E"|"F"|"a"|"b"|"c"|
"d"|"e"|"f"}("H"|"h").
```

I.6.3. (Произвольный стиль)

```
$двоичное="2#"{/"0"|"1"/}.
$восьмеричное="8#"{/"0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"/}.
$десятичное=["10#"]{/цифра/}.
$шестнадцатеричное="16#"{/цифра|"A"|"B"|"C"|"D"|"E"|
"F"|"a"|"b"|"c"|"d"|"e"|"f"/}.
```

I.7. Организация действительных чисел

Действительное число определяется следующим образом:

Варианты:

I.7.1.

```
$числовая_строка={/цифра/}.
$порядок=("E"|"e")["+"|"-"]числовая_строка.
$действительное=числовая_строка порядок|
        числовая_строка"."[числовая_строка][порядок]|
        "."числовая_строка[порядок].
```

I.7.2.

```
$числовая_строка={/цифра/}.
$порядок=("E"|"e")["+"|"-"]числовая_строка.
```

```
$действительное=

числовая_строка"."числовая_строка[порядок]|

числовая_строка порядок.
```

II. Правила, используемые в синтаксическом анализаторе

Рассматриваемые ниже правила используются при построении распознавателя. Альтернативные варианты позволяют задать структуру программы, ее операторов и выражений. Ключевые слова при описании правил выделены жирным шрифтом. Их реальное написание определяется совокупностью условий и ограничений, определяемых вариантом задания (прописные или строчные и т.д.)

II.1. Организация программы

Задается один из трех вариантов. Первый вариант определяет программу как список чередующихся описаний и операторов, разделяемых точкой с запятой. Конец текста программы определяется концом файла. Описания можно вводить непосредственно перед использованием переменных. При втором варианте программа состоит из двух независимых областей: описаний и операторов. За лексемой, определяющей конец программы, может следовать произвольная цепочка символов, так как осуществлять разбор дальше в соответствии с синтаксисом не имеет смысла. С организацией программы непосредственно связывается и структура составного оператора, синтаксис которого выдержан в аналогичном стиле. Третий вариант отличается от двух предыдущих отсутствием составного оператора.

Варианты:

II.1.1.

```
$программа={/(описание|оператор)";"/}.

$составной=BST {/оператор";"/} EST.
```

II.1.2.

```
$программа=[var описание{";" описание}]
BST оператор{";" оператор} EST.
$составной=BST оператор{";" оператор} EST.
```

II.1.3.

```
$программа={/(описание|оператор)";"/}.
```

II.2. Описания

Первый вариант описания по стилю близок к языку программирования Паскаль, второй напоминает описания Си, а третий - Бейсик.

Варианты:

II.2.1.

```
$описание=идентификатор {"," идентификатор}":"
[vector"["целое"]" of] тип.
$тип=integer|real.
```

II.2.2.

```
$описание=тип идентификатор ["["целое"]"] {"," иденти-
фикатор ["["целое"]"]}.
$тип=integer|real.
```

II.2.3.

```
$описание=dim идентификатор "("целое")" {"," идентифи-
катор "("целое")"}.
```

II.3. Синтаксис операторов

Два варианта набора операторов. Различия наблюдаются при описании их синтаксиса. Одинаковым для всех является синтаксис таких операторов, как пустой и перехода. Следует также отметить, что пустой оператор - это отсутствие каких бы то ни было конструкций, а не точка, которая в данном случае является компонентой метаязыка, завершающей правило. Кроме операторов здесь же приведен еще ряд конструкций языка, являющихся общими для всех вариантов.

Варианты:

II.3.1.

```
$оператор=[метка] непомеченный.

$непомеченный=составной|присваивания|перехода

условный|цикла|пустой|ввода|вывода.

$пустой=.

$перехода=goto имя_метки.

$ввода=read переменная {"," переменная}.

$вывода=write (выражение|спецификатор)

{"," (выражение|спецификатор)}.

$переменная=идентификатор ["["индекс"]"].

$индекс=идентификатор|целое.

$спецификатор=skip|space|tab.
```

II.3.2.

```
$оператор=метка непомеченный.

$непомеченный=присваивания|перехода|условный|цикла|

пустой|ввода|вывода.
```

```
$пустой=.

$перехода=goto имя_метки.

$ввода=input переменная {"," переменная}.

$вывода=print (выражение|спецификатор)

{"," (выражение|спецификатор)}.

$переменная=идентификатор ["("индекс")"].

$индекс=идентификатор|целое.

$спецификатор=skip|space|tab.
```

II.4. Оператор присваивания

Определяется вместе с выражением, которое ему присваивается, что обеспечивает согласованное восприятие языковых конструкций. Выражение может использоваться и в других операторах языка, например в операторе вывода. Существует три альтернативы для оператора присваивания и выражений: инфиксная форма, постфиксная скобочная форма и польская префиксная форма.

Варианты:

II.4.1. Инфиксная форма.

```
$присваивание=переменная ASG выражение.

$выражение=слагаемое {(EQ|NE|LT|GT|LE|GE) слагаемое}.

$слагаемое=множитель {(ADD|MIN) множитель}.

$множитель=унарное {(MULT|DIV|MOD) унарное}.

$унарное=[MIN] терм.
```

```
$терм=переменная|число|"("выражение")".

$число=целое|действительное.
```

II.4.2. Префиксная форма

```
$присваивание=ASG переменная"," выражение.

$выражение=простое_выражение|операнд.

$простое_выражение=операция выражение выражение|

"("MIN выражение")".

$операнд=переменная|целое|действительное.

$операция=MULT|DIV|MOD|ADD|MIN|EQ|NE|LT|GT|LE|GE.
```

II.4.3. Постфиксная форма

```
$присваивание=выражение переменная ASG.

$выражение=простое_выражение|операнд.

$простое_выражение="("выражение выражение операция")"|

"("выражение MIN")".

$операнд=переменная|целое|действительное.

$операция=MULT|DIV|MOD|ADD|MIN|EQ|NE|LT|GT|LE|GE.
```

II.5. Условный оператор

Альтернативы подобраны по стилю, обеспечивающему совпадение с синтаксисом программы, они определяют обычный условный оператор или переключатель, обеспечивающий несколько ветвлений.

Варианты:

II.5.1.

```
$условный=if выражение then непомеченный [else непоме-
ченный ].
```

II.5.2.

```
$условный=if выражение then оператор {";" оператор}
                [else оператор {";" оператор}] end.
```

II.5.3.

```
$условный=if выражение then имя_метки1"," имя_метки2","
                              имя_метки3";"
```

(В данном виде условной конструкции если значение выражения меньше нуля, то управление передается на **имя_метки1**, если значение выражения равно нулю, то управление передается на **имя_метки2**, и если значение выражения больше нуля, то управление передается на **имя_метки3**.)

II.5.4.

```
$условный=case выражение of целое ":" непомеченный
{ or целое ":" непомеченный}[ else непомеченный].
```

II.5.5.

```
$условный=switch выражение BST
                  {/case целое ":" непомеченный";"/}
                   [ default ":" непомеченный] EST.
```

Если в двух последних вариантах используется условное выражение, то **1** означает “истина”, а **0** означает “ложь”.

II.6. Оператор цикла

Синтаксис альтернатив в вариантах подобран в стиле программы, как и в случае с условными операторами.

Варианты:

II.6.1.

$цикл=loop непомеченный exit when выражение.

II.6.2.

$цикл=while выражение do непомеченный.

II.6.3.

$цикл= loop выражение оператор {";" оператор} end.

II.6.4.

$цикл=repeat непомеченный until выражение.

II.6.5.

$цикл=for идентификатор"=" переменная_цикла to перемен-

ная_цикла [step целое] [оператор] {";" оператор}

next идентификатор.

$переменная_цикла=переменная|целое.

II.6.6.

$цикл=for идентификатор"=" переменная_цикла to

переменная_цикла do непомеченный.

$переменная_цикла=переменная|целое.

II.6.7.

```
$цикл=do имя_метки идентификатор"="целое1 ","
                целое2 "," целое3 оператор {";" оператор}
                                        имя_метки continue.
```

(**целое1** – начальное значение, **целое2** – конечное значение, **целое3** – шаг.)

Варианты тестовых наборов задач

Вариант 1.

1. Найти **НОД** двух целых чисел.

2. Дана функция $y(x)=Ax^2+Bx+C$, где **A** – количество букв в фамилии студента, **B** - количество букв в имени студента, **C** - количество букв в отчестве студента. Для функции **y(x)** составить программу построения таблицы значений функции при изменении аргумента от **L** до **R** с шагом **T**. В каждой строке выводить значения аргумента и соответствующее ему значение функции. Кроме того, в конце таблицы напечатать отдельной строкой сумму таких значений функции, которые больше числа **M**.

3. Найти сумму, разность, скалярное произведение и квадрат евклидовой нормы двух векторов в **n**-мерном пространстве.

4. Отсортировать массив по возрастанию методом поиска минимального элемента.

Вариант 2.

1. Найти **НОК** двух целых чисел.

2. Дана функция $y(x)=Ax^2+Bx+C$, где **A** – количество букв в фамилии студента, **B** - количество букв в имени студента, **C** - количество букв в отчестве студента. Для функции **y(x)** составить программу построения таблицы значений функции при изменении аргумента от **L** до **R** с шагом **T**. В каждой строке выводить значения аргумента и соответствующее ему значение функции. Кроме того, в конце таблицы напечатать отдельной строкой среднее арифметическое всех отрицательных значений функции.

3. Проверить два вектора в **n**-мерном пространстве на перпендикулярность и коллинеарность.

4. Отсортировать массив по убыванию методом поиска максимального элемента.

Вариант 3.

1. Проверить, является ли целое число простым.

2. Дана функция **y(x)=Ax^2+Bx+C**, где **A** – количество букв в фамилии студента, **B** - количество букв в имени студента, **C** - количество букв в отчестве студента. Для функции **y(x)** составить программу построения таблицы значений функции при изменении аргумента от **L** до **R** с шагом **T**. В каждой строке выводить значения аргумента и соответствующее ему значение функции. Кроме того, в конце таблицы напечатать отдельной строкой сумму таких значений функции, которые находятся на отрезке от **M** до **N**.

3. Найти индексы максимального и минимального элементов в массиве.

4. Отсортировать массив по возрастанию методом “пузырька”.

Вариант 4.

1. Найти число сочетаний из **n** элементов по **m**.

2. Дана функция **y(x)=Ax^2+Bx+C**, где **A** – количество букв в фамилии студента, **B** - количество букв в имени студента, **C** - количество букв в отчестве студента. Для функции **y(x)** составить программу построения таблицы значений функции при изменении аргумента от **L** до **R** с шагом **T**. В каждой строке выводить значения аргумента и соответствующее ему значение функции. Кроме того, в конце таблицы напечатать отдельной строкой максимальное среди значений функции, квадрат которых не превосходит заданного числа **M**.

3. Определить, является ли заданное число **x** корнем полино-

ма, коэффициенты которого заданы в массиве по возрастанию степеней.

4. Отсортировать массив по убыванию методом “пузырька”.

Вариант 5.

1. Даны целые положительные **a** и **b**. Найти $c=a^b+b^a$.
2. Дана функция $y(x)=Ax^2+Bx+C$, где **A** – количество букв в фамилии студента, **B** - количество букв в имени студента, **C** - количество букв в отчестве студента. Для функции **y(x)** составить программу построения таблицы значений функции при изменении аргумента от **L** до **R** с шагом **T**. В каждой строке выводить значения аргумента и соответствующее ему значение функции. Кроме того, в конце таблицы напечатать отдельной строкой минимальное среди значений функции, больших заданного значения **M**.
3. Найти произведение двух матриц размеров **m×n** и **n×k** (Матрица хранится в одномерном массиве по строкам).
4. Отсортировать массив по возрастанию квадратов его элементов.

Варианты индивидуальных проектов

N	I							II						T
	1	2	3	4	5	6	7	1	2	3	4	5	6	
01	1	1	1	1	1	1	1	2	1	1	1	4	2	1
02	2	2	2	1	2	2	2	1	2	1	2	1	3	2
03	1	3	1	1	2	3	1	3	3	2	3	2	5	3
04	2	4	2	1	1	1	2	1	1	1	1	1	1	4
05	1	5	1	1	2	2	1	3	3	2	2	3	3	5
06	2	1	2	1	2	3	2	2	1	1	3	1	4	1
07	1	2	1	1	1	1	1	1	2	1	1	5	6	2
08	2	3	2	1	2	2	2	3	3	2	2	2	3	3
09	1	4	1	1	1	3	1	2	2	1	3	2	2	4
10	2	5	2	1	2	1	2	3	3	2	1	3	5	5
11	1	1	1	1	2	2	1	2	1	1	2	4	6	1
12	2	2	2	1	1	3	2	1	2	1	3	1	2	2
13	1	3	1	1	2	1	1	3	3	2	1	2	7	3
14	2	4	2	1	1	2	2	1	1	1	2	3	3	4
15	1	5	1	1	2	3	1	3	3	2	3	3	7	5
16	2	1	2	1	2	1	2	2	1	1	1	1	2	1
17	1	2	1	1	1	2	1	1	2	1	2	5	3	2
18	2	3	2	1	2	3	2	3	3	2	3	2	5	3
19	1	4	1	1	1	1	1	2	2	1	1	4	4	4
20	2	5	2	1	2	2	2	3	3	2	2	3	3	5
21	1	1	1	1	2	3	1	2	1	1	3	4	4	1
22	2	2	2	1	1	1	2	1	2	1	1	1	6	2
23	1	3	1	1	2	2	1	3	3	2	2	2	3	3
24	2	4	2	1	1	3	2	1	1	1	3	1	1	4

25	1	5	1	1	2	1	1	3	3	2	1	3	5	5
26	2	1	2	1	2	2	2	2	1	1	2	1	6	1
27	1	2	1	1	1	3	1	1	2	1	3	5	2	2
28	2	3	2	1	2	1	2	3	3	2	3	2	5	3
29	1	4	1	1	2	2	1	2	2	1	2	2	1	4
30	2	5	2	1	2	3	2	3	3	2	1	3	7	5
31	1	1	1	1	1	1	1	2	1	1	3	1	4	1
32	2	2	2	1	1	2	2	1	2	1	2	5	3	2
33	1	3	1	1	2	3	1	3	3	2	1	2	7	3
34	2	4	2	1	1	1	2	1	1	1	3	3	1	4
35	1	5	1	1	2	2	1	3	3	2	2	3	3	5
36	2	1	2	1	2	3	2	2	1	1	1	4	6	1
37	1	2	1	1	1	1	1	1	2	1	3	1	2	2
38	2	3	2	1	2	2	2	3	3	2	2	2	5	3
39	1	4	1	1	1	3	1	2	2	1	1	4	2	4
40	2	5	2	1	2	1	2	3	3	2	3	3	5	5
41	1	1	1	1	2	2	1	2	1	1	2	1	2	1
42	2	2	2	1	1	3	2	1	2	1	1	5	6	2
43	1	3	1	1	2	1	1	3	3	2	3	2	3	3
44	2	4	2	1	1	2	2	1	1	1	2	1	4	4
45	1	5	1	1	2	3	1	3	3	2	1	3	7	5
46	2	1	2	1	2	1	2	2	1	1	3	4	4	1
47	1	2	1	1	1	2	1	1	2	1	2	1	3	2
48	2	3	2	1	2	3	2	3	3	2	1	2	7	3
49	1	4	1	1	1	1	1	2	2	1	3	2	1	4
50	2	5	2	1	2	2	2	3	3	2	2	3	3	5
51	1	1	1	1	1	3	1	2	1	1	1	4	4	1
52	2	2	2	1	2	1	2	1	2	1	2	5	3	2

53	1	3	1	1	1	2	1	3	3	2	3	2	5	3
54	2	4	2	1	2	3	2	1	1	1	1	1	1	4
55	1	5	1	1	1	1	1	3	3	2	2	3	7	5
56	2	1	2	1	2	2	2	2	1	1	3	1	2	1
57	1	2	1	1	1	3	1	1	2	1	1	1	6	2
58	2	3	2	1	2	1	2	3	3	2	2	3	3	3
59	1	4	1	1	1	2	1	2	2	2	3	2	3	4
60	2	5	2	1	2	3	2	3	3	2	1	2	5	5
61	1	1	1	1	1	1	1	2	1	1	2	4	6	1
62	2	2	2	1	2	2	2	1	2	1	3	2	5	2
63	1	3	1	1	1	3	1	3	3	2	1	2	7	3
64	2	4	2	1	2	1	2	1	2	1	2	4	2	4
65	1	5	1	1	1	2	1	3	3	2	3	3	3	5
N	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**	**4**	**5**	**6**	**T**
	I							**II**						

Литература

5. Ахо А., Ульман Д. Теория синтаксического анализа, перевода и компиляции. – М.: Мир, 1978.
6. Бек Л. Введение в системное программирование. – М.: Мир, 1988.
7. Григорьев В. Л. Микропроцессор i486. Архитектура и программ-мирование (в 4-х книгах). Т. 1-2. ГРАНАЛ, 1993.
8. Грис Д. Построение компиляторов для цифровых вычислительных машин. – М.: Мир, 1975.
9. Квиттнер П. Задачи, Программы, Вычисления, Результаты. – М.: Мир, 1980.
10. Кауфман В. Ш. Языки программирования. Концепции и принципы. – М.: Радио и связь, 1993.
11. Скэнлон Л. Программирование на языке ассемблера. – М.: Радио и Связь, 1989.
12. Хантер Д. Проектирование и конструирование компиляторов. – М.: Финансы и статистика, 1984.
13. Вирт Н. Алгоритмы+структуры данных=программы. – М.: Мир, 1985.
14. Ахо А., Сети Р., Ульман Д. Компиляторы: принципы технологии и инструменты. Пер. с англ. – М.: Издательский дом “Вильямс”, 2003. – 768 с.
15. Серебряков В. А., Галочкин М. П. Основы конструирования компиляторов.
16. Легалов А. И. Основы разработки трансляторов (конспект лекций КГТУ). http://www.softcraft.ru

Учебное издание

Цыганова Ю.В.
Языки программирования и методы трансляции.
Лабораторный практикум.
Раздел: Методы трансляции языков программирования
Учебно-методическое пособие

Директор издательского центра
Редактирование и подготовка оригинал-макета

Подписано в печать
Формат
Усл. печ. л.
Уч.-изд. л.
Тираж 100 экз.
Заказ

Оригинал-макет подготовлен в Издательском центре
Ульяновского государственного университета

Отпечатано в издательском центре
Ульяновского государственного университета
432000, г.Ульяновск, ул. Л.Толстого, 42