УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«Брестский государственный университет имени А.С. Пушкина»
Кафедра алгебры, геометрии и математического моделирования

Д.В. Грицук, А.А. Трофимук

КОМПЬЮТЕРНАЯ АЛГЕБРА

Учебно-методический комплекс для студентов специальности 1-02 05 01 «Математика и информатика» физико-математического факультета

Брест, 2018

УДК 511+512(076)
ББК 22.13+22.14я73

Авторы:

Грицук Дмитрий Владимирович — доцент кафедры алгебры, геометрии и математического моделирования Учреждения образования «Брестский государственный университет имени А.С. Пушкина», кандидат физико-математических наук

Трофимук Александр Александрович — доцент кафедры фундаментальной и прикладной математики Учреждения образования «Гомельский государственный университет имени Франциска Скорины», кандидат физико-математических наук, доцент

Рецензенты:

Будько Александр Евгеньевич — поректор по научной работе Учреждения образования «Брестский государственный университет имени А.С. Пушкина», кандидат физико-математических наук, доцент

Кафедра информатики Учреждения образования «Гомельский государственный технический университет имени П.О. Сухого»

Компьютерная алгебра : учебно-методический комплекс для студентов специальности 1-02 05 01 «Математика и информатика» физико-математического факультета / Д.В. Грицук, А.А. Трофимук ; Брест. гос. ун-т им. А.С. Пушкина, каф. АГиММ – Брест : электронное издание БрГУ, 2018. – 270 с.

Учебно-методический комплекс написан в соответствии с действующей учебной программой по дисциплине «Компьютерная алгебра». Предназначен для студентов специальности 1-02 05 01 «Математика и информатика» физико-математического факультета.

Стиль изложения выбран таким образом, что в каждом последующем параграфе сначала даются теоретические сведения, а на их основе в дальнейшем происходит решение практических задач в системе компьютерной алгебры GAP. Опыт проведения лекционных и лабораторных занятий в течение нескольких лет показал достаточную эффективность такого подхода.

В практической части приводятся задания лабораторных работ, выполнение которых будет способствовать формированию знаний, умений и навыков в области алгоритмически разрешимых алгебраических задач и проблем.

Вспомогательный раздел содержит список использованной и рекомендованной литературы. В ЭУМК приводятся тесты для самоконтроля по курсу дисциплины «Компьютерная алгебра», а также список вопросов для подготовки к зачету.

Электронный учебно-методический комплекс ставит своей целью облегчить самостоятельную работу студентов с теоретическим материалом при подготовке к лекциям, лабораторным занятиям и зачету.

СОДЕРЖАНИЕ

Предисловие

Электронный учебно-методический комплекс написан для факультативной дисциплины «Компьютерная алгебра», читаемой на специальности «Математика и информатика». Однако, данный ЭУМК может быть полезен и студентам других специальностей физико-математического факультета Учреждения образования «Брестский государственный университет имени А. С. Пушкина». Электронное издание рассчитано на студентов, имеющих подготовку по дисциплинам, касающихся основ программирования с использованием алгоритмических языков, с целью более глубокого понимания и усвоения разделов алгебры и теории чисел.

Компьютерная алгебра — одна из областей математики и информатики, особенно активно развивающаяся в последние годы. Термин «компьютерная алгебра» объясняется способностью компьютеров манипулировать математическими выражениями, заданными символьно, а не численно. Область компьютерной алгебры охватывает алгоритмы символьных преобразований, связанных с такими абстрактными структурами, как группы, кольца и поля.

С введением данной факультативной дисциплины возникла проблема выбора средств обеспечения дисциплины, в частности систем компьютерной алгебры. Использование системы компьютерной алгебры GAP в качестве средства обучения обусловлено рядом причин: язык программирования, внешне напоминающий Pascal; стандартные типы основных

алгебраических объектов: групп (подстановок, абстрактных, матричных), колец, полей; удобные типы переменных, в т.ч. оперативно списки и записи; более 4 тыс. библиотечных функций; большая библиотека данных, включающая практически все группы, порядок которых не превосходит 2 000; бесплатное получение по сети Internet вместе с исходными текстами, являющимися наглядным пособием для освоения GAP; работа в операционных системах DOS, Windows, Unix, Linux, MacOS; работа с процессором 386 и выше с ОЗУ от 8 Mb; занимаемое место на диске — от 10 до 100 Mb в зависимости от объема инсталляции.

Все вышеизложенное привело к следующей структуре теоретической части электронного издания. В первом раздела даны краткие сведения о системе компьютерной алгебры GAP, структуре и языке программирования этой системы. Во втором разделе подробно рассматриваются вопросы, связанные с заданием группы и изучением ее свойств и строения посредством системы GAP. Третий раздел ЭУМК посвящен основным алгоритмам теории чисел. Рассматриваются элементы теории делимости и сравнимости в кольце целых чисел. На этой основе решается ряд исключительно важных для практики задач: нахождение наибольшего общего делителя, разложение на простые множители, решение сравнений и систем сравнений с одной неизвестной. В четвертом, пятом и шестом разделах рассмотрены элементы линейной алгебры, теории колец и основы криптографии. Подробно рассматриваются вопросы связанные с решением систем линейных уравнений различными способами и шиф-

рованием информации методом RSA. Заметим, что порядок изложения теоретического материала отличается от порядка предложенного в программе. По мнению авторов такая последовательность делает материал более доступным для понимания.

Авторы выбрали систему изложения, при которой в каждом последующем параграфе сначала даются теоретические сведения, а на их основе в дальнейшем происходит решение практических задач в системе компьютерной алгебры GAP. Опыт проведения лекционных и лабораторных занятий в течение нескольких лет показал достаточную эффективность такого подхода.

В практической части приводятся задания лабораторных работ, выполнение которых будет способствовать формированию знаний, умений и навыков в области алгоритмически разрешимых алгебраических задач и проблем.

Вспомогательный раздел содержит список использованной и рекомендованной литературы. В ЭУМК приводятся тесты по курсу дисциплины «Компьютерная алгебра», а также список вопросов для подготовки к зачету.

Электронный учебно-методический комплекс ставит своей целью облегчить самостоятельную работу студентов с теоретическим материалом при подготовке к лекциям, лабораторным занятиям и зачету.

При изложении теоретических вопросов алгебры и теории чисел авторы использовали издания В.С. Монахова и А.В. Бузланова [11; 12],

компьютерной алгебры — пособием А.Б. Коновалова [6].

Авторы будут благодарны читателям за отзывы, критические замечания, предложения и новые задачи. Их можно отправлять по электронной почте или по адресу: 224016, г. Брест, бульвар Космонавтов, 21.

Грицук Дмитрий Владимирович
dmitry.gritsuk@gmail.com

Трофимук Александр Александрович
alexander.trofimuk@gmail.com

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Раздел 1. ЗНАКОМСТВО С СИСТЕМОЙ КОМПЬЮТЕРНОЙ АЛГЕБРЫ GAP

Краткая характеристика и история системы GAP. Обзор возможностей GAP. Запуск системы и первые шаги. Язык программирования GAP: символы и ключевые слова в GAP, выражения, обращения к функциям, вызов процедуры, команды IF, циклы WHILE, REPEAT, FOR. Структуры данных в GAP: списки, множества, векторы и матрицы, записи.

Раздел 2. ГРУППЫ В GAP

Задание группы. Простейшие свойства группы. Силовские подгруппы. Другие виды подгрупп. Классы сопряженных элементов и сопряженных подгрупп группы. Образующие и определяющие отношения. Циклические и абелевые группы. Группы подстановок. Прямое и полупрямое произведение групп. Конечные простые группы.

Раздел 3. КОЛЬЦА И ПОЛЯ В GAP

Кольцо классов вычетов. Кольцо многочленов. Разложение многочленов на неприводимые множители. Минимальный многочлен. Введение в теорию Галуа.

Раздел 4. ВВЕДЕНИЕ В АЛГЕБРАИЧЕСКУЮ ТЕОРИЮ КОДИРОВАНИЯ

Арифметические функции. Сравнения: решение сравнений с одним неизвестным, китайская теорема об остатках. Функция Эйлера и ее приложения. Метод RSA.

ПРИМЕРНЫЙ ТЕМАТИЧЕСКИЙ ПЛАН

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Название раздела, темы	Количество часов
Раздел 1. Знакомство с системой компьютерной алгебры GAP	*2*
Краткая характеристика и история системы GAP. Обзор возможностей GAP. Запуск системы и первые шаги.	2
Язык программирования GAP: символы и ключевые слова в GAP, выражения, обращения к функциям, вызов процедуры, команды IF, циклы WHILE, REPEAT, FOR. Структуры данных в GAP: списки, множества, векторы и матрицы, записи.	2
Раздел 2. Группы в GAP	*4*
Задание группы. Простейшие свойства группы. Силовские подгруппы. Другие виды подгрупп. Классы сопряженных элементов и сопряженных подгрупп группы.	2
Образующие и определяющие отношения. Циклические и абелевые группы. Группы подстановок. Прямое и полупрямое произведение групп. Конечные простые группы.	2
Раздел 3. Кольца и поля в GAP	*4*
Кольцо классов вычетов. Кольцо многочленов. Разложение многочленов на неприводимые множители.	2
Минимальный многочлен. Введение в теорию Галуа.	2

Название раздела, темы	Количество часов
Раздел 4. Введение в алгебраическую теорию кодирования	***2***
Арифметические функции. Сравнения: решение сравнений с одним неизвестным, китайская теорема об остатках. Функция Эйлера и ее приложения. Метод RSA.	2
ИТОГО:	***14***

ПРАКТИЧЕСКАЯ ЧАСТЬ

Лабораторная работа	Количество часов
Лабораторная работа №1. Основы работы с ситемой GAP	2
Лабораторная работа №2. Списки. Целые числа. НОД целых чисел. Арифметические функции	4
Лабораторная работа №3. Функции. Команда IF, циклы FOR и WHILE	4
Лабораторная работа №4 Структура и свойства группы	4
Лабораторная работа №5 Шифрование информации методом RSA	6
ВСЕГО:	***20***

РАЗДЕЛ 1
СИСТЕМА КОМПЬЮТЕРНОЙ АЛГЕБРЫ GAP

1.1 Краткая характеристика, история и обзор возможностей системы GAP

Разработка системы компьютерной алгебры GAP (Groups, Algorithms and Programming) была начата в 1986 году в г.Аахен (Германия). В 1997 году центр координации и технической поддержки пользователей переместился в г.Сент-Эндрюс (Шотландия). В настоящее время GAP является уникальным всемирным совместным научным проектом, объединяющим специалистов в области алгебры, теории чисел, математической логики, информатики и других наук из разных стран мира.

Изначально система GAP разрабатывалась под Unix, а затем была портирована для работы в Mac OS и Windows. Однако, ряд пакетов, расширяющих функциональность системы, работает только в среде Unix/Lunix.

GAP является свободно распространяемой, открытой и расширяемой системой. Система поставляется вместе с исходными текстами, которые написаны на двух языках: ядро системы написано на Си, а библиотека функций – на специальном языке, также называемом GAP, который по синтаксису напоминает Pascal, однако является объектно-ориентированным языком. Пользователи могут создавать свои собствен-

ные программы на этом языке, и здесь исходные тексты могут служить наглядным пособие. Более того, разработчики программ для GAP могут оформить свои разработки в виде пакета для системы GAP и представить их на рассмотрение в Совет GAP. После прохождения процедуры рецензирования и одобрения советом GAP такой пакет включается в приложение к дистрибутиву GAP и распространяется вместе с ним.

Система GAP состоит из следующих четырех компонент:

- *ядро системы*, обеспечивающее поддержку языка GAP, работу с системой в программном и интерактивном режиме;
- *библиотека функций*, в которой реализованы разнообразные алгоритмы (более 4000 пользовательских функций);
- *библиотека данных*, которая включает, например, библиотеку всех групп порядка не более 2000 (за исключением 49487365422 групп порядка 2014), библиотеку примитивных групп подстановок, таблицы характеров конечных групп и т.д., что в совокупности составляет эффективное средство для выдвижения и тестирования научных гипотез;
- *обширная документация*, доступная в форматах .txt, .pdf и .html, а также через Интернет.

Изначально система компьютерной алгебры GAP (Groups, Algorithms and Programming) изначально была задумана как инструмент комбина-

торной теории групп – раздела алгебры, изучающего группы, заданные порождающими элементами и определяющими соотношениями. Однако, в дальней с выходом новых версий системы сфера ее применения охватывала все новые и новые разделы алгебры. В настоящее время выпущенна версия GAP 4.8.10 (https://www.gap-system.org/Download/index.html).

GAP дает возможность производить вычисления с гигантскими целыми и рациональными числами, допустимые значения которых ограничены только объемом доступной памяти. Более того система работает с циклотомическими полями, конечными полями, p-адическими числами, многочленами от многих переменных, рациональными функциями, векторами и матрицами. Пользователю доступны различные комбинаторные функции, элементарные теоретико-числовые функции, разнообразные функции для работы с множествами и списками.

Группы могут быть заданы как группы подстановок, матричные группы, группы, заданные порождающими элементами и определяющими соотношениями. Ряд групп может быть задан непосредственным обращением к библиотечным функциям (например, симметрическая и знакопеременная группы, группа диэдра, циклическая группа и др.).

Функции для работы с группами включают определение порядка группы, вычисление классов сопряженных элементов, центра и коммутанта группы, верхнего и нижнего центральных рядов, ряда коммутантов, Силовских подгрупп, максимальных подгрупп, нормальных подгрупп, решеток подгрупп, групп автоморфизмов, и т.д. Для ряда конечных групп

доступно определение их типа изоморфизма.

Также в системе GAP есть функция StructureDescription для вычисления структурных описаний конечных групп.

Приведенный обзрор является далеко не полным перечнем возможностей системы компьютерной алгебры GAP.

1.2 Начало работы в GAP

После запуска системы компьютерной алгебры GAP на экране появится эмблема GAP. После нее отображается дополнительная информация о версии системы и установленных компонентах:

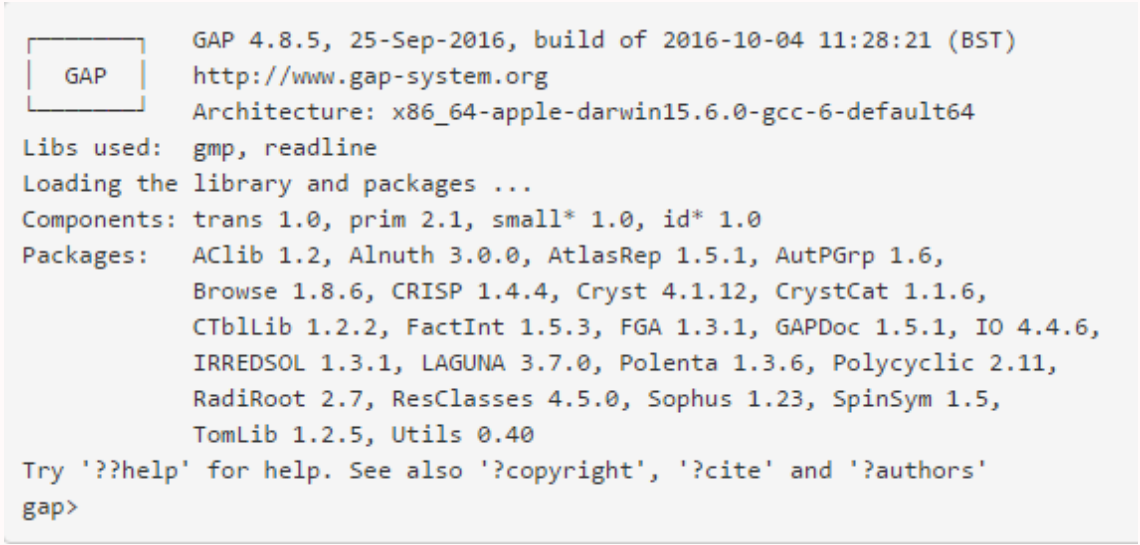
Приглашение системы (командная строка) имеет следующий вид:

```
gap>
```

Все команды в GAP заканчиваются точкой с запятой «;». Если в конце строки поставить «;;», то вычисляемое значение запишется в переменную last, но не будет выведено.

Одна команда может занимать несколько строк, последняя из которых заканчивается точкой с запятой. Таким образом, если Вы забыли поставить точку с запятой в конце строки и уже нажали клавишу *Enter*, Вы можете поставить точку с запятой в конце новой строки, а затем нажать клавишу *Enter* еще раз.

При некоторых ошибках на экран выводится промежуточное пригла-



шение системы вида brk>. Для выхода из него нужно ввести команду quit (в этом случае она не приводит к завершению работы системы).

Для просмотра истории команд используются клавиши перемещения курсора вверх и вниз. Если в командной строке набрать какой-нибудь символ, а затем нажимать клавиши перемещения курсора вверх и вниз, то будут появляться только те из ранее введенных команд, которые начинались с этого символа.

Для дублирования введенных команд и выводимых на экран результатов в текстовом файле используется команда LogTo("filename.log");. Ведение файла протокола может быть остановлено командой LogTo();

(например чтобы просмотреть его содержимое в другом окне Windows, не прерывая сеанса работы с GAP).

GAP можно использовать как простейший калькулятор:

```
gap> 3 ^ 129;
3537055373321574951456261858423755599703463477682752332729088 3
```

Каждое вычисляемое значение GAP записывает в переменную last.

Пример 1.2.1. Определите последние пять из 22 338 618 цифр 48-го числа Мерсенна $2^{74207281}-1$, найденного в январе 2016 г. и являющегося на сегодня самым большим из известных науке простых чисел:

```
gap> 2^74207281-1;
«an integer too large to be printed»
gap> last mod 100000;
36351
```

Из командной строки GAP можно вызвать подстрочную справку по встроенным функциям GAP, например если набрать в командной строке ?Factorial (без точки с запятой) для отображения справки по данной функции.

1.3 Язык программирования GAP

1.3.1 Символы и категории слов в GAP

GAP воспринимает цифры, буквы латинского алфавита (верхний и нижний регистры), пробел, символы табуляции и новой строки, а также специальные символы:

"	‘	(	)	*	+	,	-	#
.	/	:	;	<	=	>	~	&
[	\	]	^	_	{	}	!	

Составленные из символов слова можно разделить на пять категорий.

1. Ключевые слова — зарезервированные последовательности букв. Ключевыми являются следующие слова:

and	do	elif	else	end	fi
for	function	if	in	local	mod
not	od	or	repeat	return	then
until	while	quit	QUIT	break	rec
continue					

2. Идентификаторы. Идентификаторы состоят из букв, цифр, символов подчеркивания (_) и должны содержать не менее одной буквы или символа подчеркивания. При этом регистр является существенным. Идентификаторы не должны совпадать с ключевыми словами. Примеры

идентификаторов:

x	index_36	LongIdentifier
index	Index	INDEX
x5	5x	_36

3. Строки. Строками в GAP являются последовательности произвольных символов, заключенные в двойные кавычки.

4. Целые числа (последовательности цифр).

5. Знаки операций и ограничители:

+	-	*	/	^	~	!.
=	<>	<	<=	>	>=	![
:=	.	..	->	,	;	!{
[	]	{	}	(	)	:

Все вычисления и преобразования данных записываются в виде выражений. Обычно выражение включает несколько операций, которые выполняются в порядке их приоритетности.

В GAP различают арифметические и логические операции, а также операции отношений.

Арифметические операции: + (сложение), - (вычитание), * (умножение), / (деление), mod (остаток целочисленного деления), ^ (возведение в степень или сопряжение). Результат операции, как правило, зависит от типа операндов. Операция mod определена только для целых и рациональных чисел. Для элемента группы знак ^ означает возведение в

степень, если правый операнд — целое число, а если он элемент группы, то сопряжение с его помощью.

Приоритет арифметических операций (по убыванию):

1) ^;

2) *, / , mod;

3) +, -.

Операции отношения: > (больше), < (меньше), = (равно), <> (не равно), >= (не меньше), <= (не больше), in (принадлежность). Эти операции применяют к числам, символам, символьным строкам и некоторым другим структурам данных GAP. Их результатом является значение логического типа. Операции отношения = и <> проверяют, соответственно, равенство и неравенство, возвращая значение true или false. Заметим, что с их помощью можно сравнивать любые объекты, т.е. при использовании = и <> никогда не будет получено сообщение об ошибке:

```
gap> 3<>7;
true
gap> "seven-7;
false
gap> 3<>[1,2,3];
true
```

Логические операции: and (и), or (или), not (не). Эти операции выполняют с логическими переменными и константами. Результатом вы-

полнения логической операции является значение логического типа.

1.3.2 Структуры данных в GAP

Константы

Задание числовых констант:

```
gap> 4938270/75;
987654/15
gap> -5; 29 - 31;
-5
-2
gap> 5^55;
27755575615628913510590791702270507812 5
```

Задание символьных констант:

```
gap> 'n';
'n'
gap> "Hello!";
"Hello!"
```

Списки

Список — это заключенный в квадратные скобки набор объектов, разделенных запятыми. Например, список из первых десяти простых нечетных чисел можно задать следующим образом:

```
gap> primes:=[3, 5, 7, 11, 13, 17, 19, 23, 29, 31];
[3, 5, 7, 11, 13, 17, 19, 23, 29, 31 ]
```

Затем к нему можно добавить следующие три простых числа:

```
gap> Append(primes, [37, 41, 43]);
gap> primes;
[3, 5, 7, 11, 13,17, 19, 23, 29, 31, 37, 41, 43]
```

Добавить один элемент можно и по-другому:

```
gap> Add(primes, 47);
gap> primes;
[ 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47 ]
```

Указать отдельный элемент списка можно по его номеру в списке:

```
gap> primes[8];
23
```

Этот же механизм позволяет присвоить значение существующему или

новому элементу списка (функция Length определяет длину списка):

```
gap> Length(primes);
14
gap> primes[15]:= 53;
53
gap> primes;
[ 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53 ]
```

При этом значение не обязательно должно присваиваться следующему элементу списка. Например, если девятнадцатым простым нечетным числом является 71, мы можем сразу присвоить значение 71 девятнадцатому элементу списка primes, пропуская недостающие элементы. Полученный список будет иметь длину 19:

```
gap> primes[19]:= 71;
71
gap> primes;
[ 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,,, 71 ]
gap> Length(primes);
19
```

Список должен быть создан перед заданием его элемента (например

быть пустым списком [ ]):

```
gap> sp[1]:= 3;
Error, Variable: 'sp' must have a value
gap> sp:= [];;
gap> sp[1]:= 3;
3
```

Функция Position возвращает номер первого элемента списка, имеющего заданное значение. Если в списке нет элемента с заданным значением, функция возвращает fail:

```
gap> Position(primes, 3);
7
gap> Position(primes, 2);
fail
```

Заметим, что при всех приведенных выше изменениях списка primes длина списка изменялась автоматически. Функция IsBound для списков показывает, содержит ли список элемент с заданным номером (для

записей — содержит ли запись указанное поле):

```
gap> l:= [ 1, 2, , , 5, 6, ,8 , , , 11 ];;
gap> IsBound(l[3]);IsBound(l[5]); IsBound(k[18]);
false
true
false
```

Список может состоять из объектов различных типов, например:

```
gap> sp:= [true, "five" ,, 5];
[ true, "five" ,, 5 ]
```

Список может являться частью другого списка:

```
gap> sp[3]:= [1, 2, 3, 4, 5];;
gap> sp;
[ true, "five" , [1, 2, 3, 4, 5 ],, 5 ]
```

Извлечение и изменение подмножеств списка производит оператор { }. Например:

```
gap> sp_new := sp{ [ 1, 2, 3 ] };
[ true, "five", [ 1, 2, 3, 4, 5] ]
gap> sp_new[2,3]:=["seven",[1, 2, 3, 4, 5, 6, 7]];
[ "seven", [ 1, 2, 3, 4, 5, 6, 7 ] ]
gap> sp_new;
```

```
[ true, "seven", [ 1, 2, 3, 4, 5, 6, 7 ] ]
```

Для работы со списками используются также следующие встроенные функции:

• List(list,func) возвращает новый список new, в котором каждый i-й элемент является результатом выполнения функции func(list[i]), т.е. new[i] = func(list[i]).

```
gap> List( [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], i -> i^2 );
[ 1, 4, 9,16,25,36, 49, 64, 81, 100 ]
gap> List( [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], IsPrime );
[ false, true, true, false, true, false, true, false, false, false ]
```

• Filtered(list, func) возвращает новый список, содержащий элементы списка list, удовлетворяющие условию func.

```
gap> Filtered( [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], IsPrime );
[ 2, 3, 5, 7 ]
```

Строки

Строки являются частным случаем списков и печатаются без разделителей. Примеры задания строк и операций над ними:

```
gap> text1:=['I',' ','l','o','v','e',' ','m','a','t','h','!'];
"I love math!"
```

```
gap> text2 := "I love math!";
"I love math!"
gap> text1 = text2;
true
gap> text2[8];
'm'
```

Арифметические прогрессии

Целочисленные конечные арифметические прогрессии являются специальным видом списков. Они описываются первым, вторым и последним элементами, разделенными соответственно запятой или двумя точками и заключенными в квадратные скобки. Если прогрессия состоит из последовательных чисел, второй элемент может быть опущен.

```
gap> [1..100]; #натуральные числа от 1 до 100
[1 .. 100 ]
gap> [1,2..100]; #эквивалентно предыдущей команде
[1 .. 100 ]
gap> [5,10..100]; # здесь шаг равен 5
[ 5,10 .. 100 ]
```

```
gap> Length( last );
20
gap> [ 100, 95 .. 5];
[ 100, 95 .. 5 ]
```

Множества

Множествами в GAP называются списки специального вида. Элементы множества расположены последовательно, упорядочены (порядок сортировки GAP определяет самостоятельно) и встречаются в списке только один раз. Множества, как и списки, могут содержать объекты различных типов.

Проверить, является ли объект множеством, можно с помощью функции IsSet. Для каждого списка существует соответствующее ему множество, получаемое с помощью функции Set.

```
gap> auto:=["bmw", "mercedes", "volkswagen", "ford", "audi",
"bmw"];;
gap> IsSet(auto);
false
gap> auto:=Set(auto);
[ "audi", "bmw", "ford", "mercedes", "volkswagen"]
```

Заметим, что при этом исходный список auto был изменен.

Для проверки принадлежности объекта множеству используется оператор in. Его также можно использовать для проверки принадлежности к списку, однако в первом случае проверка выполняется быстрее, т.к. сортировка позволяет использовать двоичный поиск вместо последовательного перебора.

```
gap> "bmw"in auto;
true
gap> "gelly"in auto;
false
```

Добавить к множеству новый элемент можно с помощью функции AddSet (обратите внимание на порядок следования элементов):

```
gap> AddSet(auto, "gelly");
gap> auto;
[ "audi", "bmw", "ford", "geely", "mercedes", "volkswagen"]
gap> AddSet(auto, "bmw");
gap> auto; # 'auto' не изменилось
[ "audi", "bmw", "ford", "geely", "mercedes", "volkswagen"]
```

Пересечение, объединение и разность множеств определяются с помощью функций Intersection, Union и Difference. При этом аргументы могут быть обычными списками, тогда как результат всегда будет являться множеством. Те же операции над множествами производят функ-

ции IntersectSet, UniteSet и RemoveSet, но они не возвращают результат, а заменяют им первый аргумент.

Векторы и матрицы

Вектор является не содержащим пробелов списком элементов, принадлежащих общему полю.

```
gap> v:= [3, 4,5/2];
[3, 4, 5/2]
gap> IsRowVector(v);
true
```

Векторы умножаются на скаляры из любого поля, содержащего данное. Умножение двух векторов равной длины дает их скалярное произведение.

```
gap> 2 * v;
[ 6, 8, 5]
gap> v * 1/5; # это эквивалентно команде v/5;
[ 6/5, 8/5, 1 ]
gap> v * v; # скалярное произведение v на себя
5
```

Матрица — список векторов одинаковой длины, не содержащий про-

белов:

```
gap> M:= [[1, 2, 3],
> [4, 5, 6],
> [7, 8, 9]];
[ [ 1, 2, 3 ], [ 4, 5, 6 ], [7 , 8, 9 ] ]
gap> m[3][2];
8
```

Матрицы можно умножать на скаляры, векторы и другие матрицы (при этом умножение обобщается и возможно также при несоответствии размеров):

```
gap> Display(M);
[ [ 1, 2, 3 ],
[ 4, 5, 6 ],
[ 7, 8, 9 ] ]
gap> [1,0,0]*M;
[ 1, 2, 3 ]
gap> [1,0,0,0]*M;
[ 1, 2, 3 ]
gap> M*[1,0,0];
[ 1, 4, 7 ]
gap> M*[1,0,0,0];
[ 1, 4, 7 ]
```

Заметим, что умножение вектора на матрицу приводит к линейной комбинации строк матрицы, тогда как умножение матрицы на вектор приводит к линейной комбинации ее столбцов. В последнем случае вектор рассматривается как вектор-столбец.

Подматрицы извлекаются или изменяются с помощью фигурных скобок { }:

```
gap> sm:=M{[1,2]}{[2,3]};
[ [ 2, 3 ], [ 5, 6 ] ]
gap> sm{[1,2]}{[2]}:=[[1],[-1]];
[ [ 1 ], [ -1 ] ]
gap> sm;
[ [ 2, 1 ], [ 5, -1 ] ]
```

Первая пара скобок указывает выбранные строки, а вторая — столбцы.

Записи

Записи — способ создания новых структур данных. Как и списки, записи — наборы других объектов (которые называются компонентами, или полями), обращение к которым происходит не по номеру, а по имени.

```
gap> date:= rec(year:=1992, month:="Jan" , day:=13);
rec( day := 13, month := "Jan" , year := 1992 )
```

Изначально запись определяется как разделенный запятыми список присваиваний значений ее полям. Для обращения к значению соответствующего поля записи необходимо указать имя записи и имя поля, разделив их точкой. Определив запись, в дальнейшем можно добавлять к ней новые поля.

```
gap> date.year;
1992
gap> date.time:=rec(hour:=19, minute:=23, second:=12);
rec ( hour := 19, minute := 23, second := 12 )
gap> date;
rec( day := 13, month := "Jan" , time := rec( hour := 19, minute
:= 23, second := 12 ), year := 1992)
```

Для определения, является ли объект записью, применяется функция IsRecord. Структуру записи можно получить с помощью функции RecNames:

```
gap> RecNames(date);
[ "time" , "year" , "month" , "day"]
```

1.3.3 Операторы

Оператор присваивания

Формат: *<имя переменной>* := *<значение переменной>* ;

```
gap> a:=3;
3
gap> b:=-2*5;
-10
gap> c:=a+b;
-7
gap> A:=[1,2,3,4];
[ 1, 2, 3, 4 ]
gap> b2:=b;
-10
```

Условный оператор IF

Формат:

if *<условие 1>* **then** *<последовательность команд 1>* ;
[**elif** *<условие 2>* **then** *<последовательность команд 2>* ;]
[**else** *<последовательность команд 3>*]
fi;

При этом частей elif может быть произвольное количество или ни одной. Часть else также может отсутствовать.

Пример 1.3.1. Определите знак числа i.

```
gap> i := 10;;
gap> if 0 < i then
> s := 1;
> elif i < 0 then
> s := -1;
> else
> s := 0;
> fi;
gap> s; # знак i
1
```

Оператор цикла с предусловием WHILE

Формат: **while** <*условие*> **do** <*последовательность команд*> **od;**

<*Последовательность команд*> выполняется, пока истинно <*условие*>. При этом сначала проверяется условие, а затем, если оно истинно, выполняются команды. Если уже при первом обращении условие ложно, то <*последовательность команд*> не выполнится ни разу.

Пример 1.3.2. Вычислите сумму квадратов первых последовательных натуральных чисел, не превышающих 200.

```
gap> i:=1;; S:=0;;
gap> while i<=200 do
> S:=S+i^2;
> i:=i+1;
> od;
gap> S;
2686700
```

Оператор цикла с постусловием REPEAT

Формат:

repeat <*последовательность команд*> **until** <*условие*> **;**

<*Последовательность команд*> выполняется до тех пор, не будет выполняться условие <*условие*>. При этом сначала команды, а затем проверяется условие. Таким образом, при любом значении условия <*последовательность команд*> , по крайней мере, один раз.

Пример 1.3.3. Вычислите сумму квадратов первых последовательных натуральных чисел, не превышающих 200.

```
gap> i := 1;; s := 0;;
gap> repeat
```

```
> s := s + i^2; i := i + 1;
> until i > 200;
gap> s;
2686700
```

Оператор цикла FOR

Формат:

for <*имя переменной*> **in** <*список значений переменной*> **do** <*последовательность команд*> **od;**

<*Последовательность команд*> выполняется для каждого элемента из списка <*список значений переменной*>.

Пример 1.3.4. Найдите сумму положительных четных чисел, не превышающих 100.

```
gap> s := 0;;
gap> for i in [2,4..100] do
> s := s + i;
> od;
gap> s;
2550
```

Пример 1.3.5. Для заданной группы найдите элемент порядка 3.

```
gap> g := Group((1,2,3,4,5),(1,2)(3,4)(5,6));
Group([(1,2,3,4,5),(1,2)(3,4)(5,6) ])
gap> for x in g do
> if Order(x) = 3 then
> break; fi; od;
gap> x;
(1,4,3)(2,6,5)
```

Команда RETURN

Формат:

return;
return <*выражение*> **;**

Первая форма прерывает выполнение внутренней (при вызове одной функции из другой) функции и передает управление вызывающей функции, не возвращая при этом никакого значения. Вторая, кроме того, возвращает значение выражения <*выражение*>.

1.3.4 Процедуры и функции

Различие между процедурами и функциями введено для удобства работы, GAP же их не различает. Функция возвращает значение, но

не производит побочных эффектов. Процедура не возвращает никакого значения, но производит какое-либо действие (например процедуры Print, Append, Sort).

Процедуры

Формат вызова процедуры:
procedure-var();
procedure-var(arg-expr , arg-expr);

Функции

Система компьютерной алгебры GAP имеет более 4 000 библиотечных функций. Формат обращения к библиотечным функциям GAP:

function-var()
function-var(arg-expr , arg-expr)

В GAP имеется возможность разработки пользовательских функций. Функция записывается в текстовый файл с расширением *.g*.

Формат определения пользовательской функции:

function ([arg-ident , arg-ident])
[local loc-ident , loc-ident ;]
<последовательность команд>
end

Для дальнейшего использования созданной функции она загружается в систему с помощью процедуры

Read("путь к файлу с расширением .g");.

Пример 1.3.6. Напишите функцию, которая определяет n-е число Фибоначчи.

```
gap> fib := function ( n )
> local f1, f2, f3, i;
> f1 := 1; f2 := 1;
> for i in [3..n] do
> f3 := f1 + f2; f1 := f2; f2 := f3;
> od;
> return f2;
> end;;
gap> List( [1..10], fib );
[ 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ]
```

РАЗДЕЛ 2
ЭЛЕМЕНТЫ ТЕОРИИ ГРУПП В СИСТЕМЕ GAP

2.1 Группы, подгруппы. Порождающие множества. Циклические группы, подгруппы циклической группы

Определение 2.1.1 Множество G с заданной на нём бинарной алгебраической операцией (умножением) называется *группой*, если:

1) эта операция ассоциативна, т.е. $(ab)c = a(bc)$ для любых элементов a, b, c из G;

2) в G существует единичный элемент e такой, что $ae = ea = a$ для любого элемента a из G;

3) для каждого элемента a из G в G существует обратный элемент a^{-1} такой, что $a^{-1}a = aa^{-1} = e$.

Все определения и результаты легко переносятся на множества с аддитивной формой записи операции.

Определение 2.1.2. Группа G называется *абелевой, или коммутативной*, если все элементы группы перестановочны между собой, т.е. выполняется коммутативный закон $ab = ba$ для любых элементов a, b из группы G.

Пример 2.1.1.

1. Множества $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}, \mathbb{C}$ с операцией сложения — абелевы группы.

2. Множество $\mathbb{N}$ со сложением не является группой, т.к. в $\mathbb{N}$ нет ну-

левого и противоположных элементов. Однако $\mathbb{N}$ со сложением — коммутативная полугруппа.

3. Множество $\{-1, 1\}$ с умножением — конечная абелева группа порядка 2.

4. Ни одно из множеств $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ с умножением группу не образует. Если положим $\mathbb{C}^* = \mathbb{C}\setminus\{0\}$, $\mathbb{R}^* = \mathbb{R}\setminus\{0\}$, $\mathbb{Q}^* = \mathbb{Q}\setminus\{0\}$, то $\mathbb{C}^*$, $\mathbb{R}^*$ и $\mathbb{Q}^*$ с умножением являются абелевыми группами. Множества $\mathbb{Z}^* = \mathbb{Z}\setminus\{0\}$ и $\mathbb{N}$ с умножением — коммутативные полугруппы с единицей, но не группы.

Определение 2.1.3. *Порядком элемента* a называется наименьшее натуральное число n такое, что $a^n = e$ и обозначается $|a|$. *Порядком группы* называется количество ее элементов. Обозначается порядок группы G через $|G|$. В случае, если множество элементов бесконечно, говорят, что G имеет бесконечный порядок, и пишут $|G| = \infty$.

Пусть $X = \{1, 2, \ldots, n\}$ и S_n — совокупность всех подстановок степени n. Множество S_n с операцией умножения образует конечную группу порядка $n!$ с единичным элементом

$$\varepsilon = \begin{pmatrix} 1 & 2 & \ldots & n \\ 1 & 2 & \ldots & n \end{pmatrix}.$$

Группу S_n называют *симметрической группой степени* n. При $n \geq 3$ эта группа неабелева.

Четные подстановки образуют конечную группу A_n порядка $n!/2$, ко-

торую называют *знакопеременной группой степени* n.

Определение 2.1.4. Подмножество H группы G называется *подгруппой*, если H — группа относительно той же операции, которая определена на G. Запись $H \leq G$ означает, что H — подгруппа группы G, а $H < G$, что H — *собственная подгруппа группы* G, т.е. $H \leq G$ и $H \neq G$.

Теорема 2.1.1 (критерий подгруппы). Непустое подмножество H группы G будет подгруппой тогда и только тогда, когда $h_1 h_2 \in H$ и $h_1^{-1} \in H$ для всех $h_1, h_2 \in H$.

Отметим, что каждая группа G обладает *единичной подгруппой* $E = \{e\}$. Сама группа G также считается подгруппой в G. Эти подгруппы называют *тривиальными подгруппами. Нетривиальная подгруппа* группы G — это такая подгруппа H из G, которая отлична от G и E.

Пример 2.1.2.

1. Т.к. $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ — аддитивные группы и $\mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$, то $\mathbb{Z} < \mathbb{Q} < \mathbb{R} < \mathbb{C}$.

2. Поскольку $\mathbb{Q}^*$, $\mathbb{R}^*$, $\mathbb{C}^*$, $\{-1, 1\}$ — мультипликативные группы, то $\{-1, 1\} < \mathbb{Q}^* < \mathbb{R}^* < \mathbb{C}^*$.

3. Т.к. S_n и A_n — группы с одной и той же операцией и $A_n \subseteq S_n$, то $A_n < S_n$.

4. Примером подгруппы группы отличных от нуля комплексных чисел по умножению могут служить все комплексные числа, являющиеся корнями n-ой степени из единицы. Еще одну подгруппу этой же группы образуют все комплексные числа, равные по абсолютной величине

единице.

Теорема 2.1.2. Произведение двух подгрупп A и B группы G является подгруппой тогда и только тогда, когда A и B перестановочны, т.е. $AB = BA$.

Определение 2.1.5. Пусть M — произвольное подмножество группы G. Пересечение всех подгрупп из G, содержащих M, называется *подгруппой, порожденной множеством* M. Множество M в этом случае называется *порождающим множеством*, и подгруппа, им порожденная, обозначается $\langle M \rangle$.

Теорема 2.1.3. Если M — подмножество группы G, то

$$\langle M \rangle = \{a_1^{m_1} a_2^{m_2} \dots a_n^{m_n} \mid a_i \in M, m_i = \pm 1, n = 1, 2, 3, \dots\}.$$

Определение 2.1.6. *Диэдральной группой* называется группа, порожденная двумя различными инволюциями, где под инволюцией понимают элемент порядка 2.

Определение 2.1.7. *Группой кватернионов* называется группа, порожденная двумя матрицами над полем комплексных чисел:

$$\begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Определение 2.1.8. Группа, порожденная одним элементом a, называется *циклической* и обозначается $< a >$.

Теорема 2.1.4. Любая подгруппа циклической группы — циклическая группа.

Пример 2.1.3. К циклической группе относится аддитивная группа $\mathbb{Z}$, порождающим элементом которой является 1. Аддитивная группа $m\mathbb{Z}$ является циклической группой, порождающим элементом которой является m, где $m \in \mathbb{N}$.

Пример 2.1.4. Выясните, будет ли подгруппой произведение групп $A = \langle (12) \rangle$ и $B = \langle (13) \rangle$ группы S_3?

Подгруппы A и B состоят из следующих элементов:

$A = \{e, (12)\}, B = \{e, (13)\}$.

Найдем произведения AB и BA:

$AB = \{e, (12)\} \cdot \{e, (13)\} = \{e, (12), (13), (132)\}$,

$BA = \{e, (13)\} \cdot \{e, (12)\} = \{e, (13), (12), (123)\}$.

Т.к. $AB \neq BA$, то по теореме 3.1.2 AB не является подгруппой группы S_3.

В 1872 г. была доказана основная для теории конечных групп теорема, описывающая строение максимальных p-подгрупп конечной группы. Теорема доказана норвежским математиком Л. Силовым. Поэтому максимальные p-подгруппы названы в его честь силовскими p-подгруппами.

Напомним, что группа, порядки всех элементов которой являются степенями некоторого фиксированного простого числа p, называется p-группой.

Определение 2.1.9. Максимальная p-подгруппа называется *силов-*

ской p-подгруппой.

Теорема 2.1.5 (Теорема Силова). Пусть G — конечная группа, p — простое число. Тогда справедливы следующие утверждения:

1) для каждой степени p^k, делящей порядок G, в G существует подгруппа порядка p^k;

2) если p^k делит порядок G, то каждая подгруппа порядка p^k из G вложена в некоторую подгруппу порядка p^{k+1} из G. В частности, силовские p-подгруппы из G — это в точности подгруппы порядка p^r, где p^r — максимальная степень p, делящая порядок G;

3) все силовские p-подгруппы из G сопряжены в G;

4) количество силовских p-подгрупп из G сравнимо с единицей по модулю p и делит порядок G.

Теорему Силова можно применять для исследования строения групп небольших порядков. С ее помощью можно определить простоту группы или найти точное количество силовских подгрупп, решать другие вопросы о строении группы.

Реализация в системе GAP

В дальнейшем все вычисления, проводимые в системе GAP, будем применять для мультипликативных групп.

Задание группы осуществляется при помощи следующих функций системы GAP:

• Group(gen1, gen2, ..., genN) — группа, порожденная элементами gen1, gen2, ..., genN;

• GroupByGenerators(gens) — группа, порожденная элементами из списка *gens*;

• GeneratorsofGroup(G) возвращает список порождающих элементов группы G;

• AsGroup(G) возвращает группу, если элементы множества G ее образуют, или возвращает «fail», если элементы множества G группу не образуют;

• SymmetricGroup(n) возвращает группу подстановок степени n;

• AlternatingGroup(n) возвращает группу четных подстановок степени n;

• ElementaryAbelianGroup(n) возвращает элементарную абелеву группу порядка n;

• DihedralGroup(n) возвращает диэдральную группу порядка n;

• QuaternionGroup(n) возвращает группу кватернионов порядка n;

• AbelianGroup(ints) возвращает абелеву группу, изоморфную группе вида $Z_{inst1} \times Z_{inst2} \times ... \times Z_{instS}$, где $ints$ — список положительных целых чисел $[inst1, inst2, ..., instS]$;

• IsSubgroup(G,H) возвращает «true», если непустое подмножество H группы G является подгруппой группы G, и «false» — в противном случае.

• Size(G) возвращает порядок группы G.

Пример 2.1.5. Будет ли группой множество:
1) $S = \{e, (13)(24), (1234), (1432)\}$;
2) $K = \{e, (12)(34), (13)(24), (14)(23)\}$;
3) $L = \{e, (23), (24), (34), (234), (243)\}$?

```
gap> S:=[(),(1,3)(2,4),(1,2,3,4),(1,4,3,2)];
[ (), (1,3)(2,4), (1,2,3,4), (1,4,3,2) ]
gap> S:=AsSet(S);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap> AsGroup(S);
Group([ (1,2,3,4) ])
gap> K:=[(),(1,2),(3,4),(1,3,2,4),(1,4,2,3)];  [ (), (1,2), (3,4),
(1,3,2,4), (1,4,2,3) ]
gap> K:=AsSet(K);
[ (), (3,4), (1,2), (1,3,2,4), (1,4,2,3) ]
gap> AsGroup(K);
fail
gap> L:=[(),(2,3),(2,4),(3,4),(2,3,4),(2,4,3)];
[ (), (2,3), (2,4), (3,4), (2,3,4), (2,4,3) ]
gap> L:=AsSet(L);
[ (), (3,4), (2,3), (2,3,4), (2,4,3), (2,4) ]
gap> AsGroup(L);
Group([ (3,4), (2,3) ])
```

Пример 2.1.6. Задайте группу подстановок, которая порождается подстановками (1,2) и (1,2,3,4,5,6), и укажите ее порядок.

```
gap> a:=(1,2);
(1,2)
gap> b:=(1,2,3,4,5,6);
(1,2,3,4,5,6)
gap> G:=Group(a,b);
Group([ (1,2), (1,2,3,4,5,6) ])
gap> GeneratorsOfGroup(G);
[ (1,2), (1,2,3,4,5,6) ]
gap> Size(G);
720
```

В системе GAP циклическую группу определенного порядка можно задать следующим образом:

• CyclicGroup([filtr],n) задает циклическую группу порядка n. Дополнительное условие [filtr] показывает в каком классе (подстановок, матриц) создается циклическая группа.

Пример 2.1.7. Задайте циклическую группу порядка 12 в классе подстановок и матриц.

```
gap> CyclicGroup(IsPermGroup,12);
Group([ (1,2,3,4,5,6,7,8,9,10,11,12) ])
```

```
gap> Size(last);
12
gap> CyclicGroup(IsMatrixGroup,12);
<matrix group of size 12 with 1 generators>
gap> Size(last);
12
```

Пример 2.1.8. В циклической группе подстановок $\langle a \rangle$ порядка 24 найдите все элементы g, удовлетворяющие условию $g^6 = e$.

```
gap> G:=CyclicGroup(IsPermGroup,24);
Group([
(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24) ])
gap> Filtered(G,i->i*i*i*i*i*i=());
[ (),
(1,21,17,13,9,5)(2,22,18,14,10,6)(3,23,19,15,11,7)(4,24,20,16,12,8),
(1,17,9)(2,18,10)(3,19,11)(4,20,12)(5,21,13)(6,22,14)(7,23,15)(8,24,16),
(1,13)(2,14)(3,15)(4,16)(5,17)(6,18)(7,19)(8,20)(9,21)(10,22)(11,23)
(12,24),
(1,9,17)(2,10,18)(3,11,19)(4,12,20)(5,13,21)(6,14,22)(7,15,23)(8,16,24),
(1,5,9,13,17,21)(2,6,10,14,18,22)(3,7,11,15,19,23)(4,8,12,16,20,24)
]
gap> Size(last);
6
```

Пример 2.1.9. Выясните, будет ли подгруппой группы S_3 произведение групп $A = \langle (12) \rangle$ и $B = \langle (13) \rangle$.

```
gap> A:=Group((1,2));
Group([ (1,2) ])
gap> B:=Group((1,3));
Group([ (1,3) ])
gap> AA:=AsList(A);
[ (), (1,2) ]
gap> BB:=AsList(B);
[ (), (1,3) ]
gap> C:=[];
[ ]
gap> for i in [1..Size(AA)] do
> for j in [1..Size(BB)] do
> Add(C,AA[i]*BB[j]);
> od;
> od;
gap> C;
[ (), (1,3), (1,2), (1,2,3) ]
gap> AsGroup(C);
fail
```

Определение 2.2.1. *Левым смежным классом* группы G по подгруппе H называется множество $xH = \{xh \mid h \in H\}$. Элемент x называется представителем смежного класса. Правый смежный класс определяется аналогично.

Свойства смежных классов:

1) смежные классы либо не пересекаются, либо совпадают;

2) смежные классы равномощны;

3) элементы a, b содержатся в одном смежном классе по подгруппе H, если $b^{-1}a \in H$ $(ba^{-1} \in H)$.

Пример 2.2.1. Определите из каких элементов состоит левый смежный класс симметрической группы S_3 по подгруппе $< (12) >$ с представителем (123).

Очевидно, что циклическая подгруппа $< (12) >$ состоит из элементов $\{e, (12)\}$. Тогда элементами левого смежного класса $(123) < (12) >$ будут $\{(123) * e = (123), (123) * (12) = (13)\}$.

Ответ: $(123) < (12) >= \{(123), (13)\}$.

Пример 2.2.2. Определите из каких элементов состоит правый смежный класс группы $GL(2, R)$ по подгруппе $H = \left\langle \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \right\rangle$ с предста-

вителем $g = \begin{bmatrix} 3 & 1 \\ 1 & -2 \end{bmatrix}$.

Очевидно, что циклическая подгруппа H состоит из элементов $\left\{ E_2, \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \right\}$. Тогда элементами правого смежного класса Hg будут $\left\{ \begin{bmatrix} 3 & 1 \\ 1 & -2 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} * \begin{bmatrix} 3 & 1 \\ 1 & -2 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ -1 & 2 \end{bmatrix} \right\}$.

Ответ: $Hg = \left\{ \begin{bmatrix} 3 & 1 \\ 1 & -2 \end{bmatrix}, \begin{bmatrix} 3 & 1 \\ -1 & 2 \end{bmatrix} \right\}$.

Пример 2.2.3. Найдите все левые смежные классы группы S_3 по подгруппе $H = \langle (12) \rangle$.

Т.к. $S_3 = \{e, (12), (13), (23), (123), (132)\}$, то левыми смежными классами по H будут множества:

$eH = (12)H = H = \{e, (12)\}$,

$(13)H = (13)\{e, (12)\} = \{(13), (123)\} = (123)H$,

$(23)H = (23)\{e, (12)\} = \{(23), (132)\} = (132)H$.

Ответ: $S_3 = H \cup (13)H \cup (23)H$.

Определение 2.2.2. Количество левых (правых) смежных классов группы G по подгруппе H называется *индексом подгруппы H в группе G* и обозначается $|G : H|$.

Теорема 2.2.1 (Теорема Лагранжа). Для любой подгруппы H конечной группы G справедливо следующее равенство:

$$|G| = |G : H||H|.$$

Следствие 2.2.1. Порядок подгруппы делит порядок группы.

Следствие 2.2.2. Порядок элемента делит порядок группы.

Следствие 2.2.3. Группа простого порядка циклическая.

Пример 2.2.4. Найдите все подгруппы симметрической группы S_3 степени 3.

Порядок $|S_3| = 6$, поэтому по теореме Лагранжа ее подгруппы могут быть только следующих порядков: 1, 2, 3, 6. Подгруппы порядков 1 и 6 — это единичная подгруппа $H_1 = \langle e \rangle$ и вся группа $H_2 = S_3$. Подгруппы порядков 2 и 3, согласно следствию 2.2.3 теоремы Лагранжа, циклические, поэтому находим все подгруппы, порожденные неединичными элементами группы S_3:

$H_3 = \langle (12) \rangle = \{e, (12)\}$,
$H_4 = \langle 13) \rangle = \{e, (13)\}$,
$H_5 = \langle (23) \rangle = \{e, (23)\}$,
$H_6 = \langle (123) \rangle = \{e, (123), (132)\}$,
$H_7 = \langle (132) \rangle = \{e, (132), (123)\}$.

Т.к. $H_6 = H_7$, то S_3 имеет в точности шесть подгрупп: H_1, H_2, H_3, H_4, H_5, H_6.

Ответ: H_1, H_2, H_3, H_4, H_5, H_6.

Определение 2.2.3. Подгруппа H *нормальна* в группе G (обозначается $H \triangleleft G$), если левые и правые смежные классы группы G по подгруппе

H совпадают.

Определение 2.2.4. Элемент a *сопряжен* с элементом b в группе G, если найдется такой x из G, что $x^{-1}ax = b$.

Кроме того, обозначение $x^{-1}ax = a^x$ переносится на множества:

$$A^B = \{a^b \mid a \in A, b \in B\},$$

где подмножество A^x называется *подмножеством, сопряженным подмножеству A посредством элемента x*.

Теорема 2.2.2. Подгруппа H нормальна в группе G тогда и только тогда, когда она совпадает с каждой своей сопряженной подгруппой.

Теорема 2.2.3. Порядки сопряженных элементов равны.

Сопряжение — отношение эквивалентности, т.е для сопряжения выполняются три свойства: рефлексивность, симметричность и транзитивность. Вся группа разбивается на непересекающиеся классы сопряженных элементов a^G. Во всех числовых системах и абелевых группах классы сопряженных элементов состоят из одного элемента. Вообще, различные классы могут иметь разные мощности. Инструментом измерения мощности класса служит нормализатор.

Пример 2.2.5. Найдите в S_3 все классы сопряженных элементов.

Т.к. $S_3 = \{e, (12), (13), (23), (123), (132)\}$, то найдем $(12)^{S_3}$. Очевидно, что $(12)^e = (12)^{(12)} = (12)$. Кроме того,

$(12)^{(13)} = (31)(12)(13) = (23)$;

$(12)^{(23)} = (32)(12)(23) = (13)$;

$(12)^{(123)} = (321)(12)(123) = (13)$;
$(12)^{(132)} = (231)(12)(132) = (23)$.

Таким образом, $(12)^{S_3} = \{(12), (23), (13)\}$. Аналогично можно убедиться, что в S_3 имеется три класса сопряженных элементов:
$e^{S_3} = \{e\}$; $(12)^{S_3} = (13)^{S_3} = (23)^{S_3} = \{(12), (23), (13)\}$;
$(123)^{S_3} = (132)^{S_3} = \{(123), (132)\}$.
Поэтому, $S_3 = e^{S_3} \cup (12)^{S_3} \cup (123)^{S_3}$.
Ответ: $S_3 = e^{S_3} \cup (12)^{S_3} \cup (123)^{S_3}$.

Определение 2.2.5. *Нормализатором* множества M в группе G называется множество $N_G(M) = \{h \mid hM = Mh, h \in G\}$.

Определение 2.2.6. *Централизатором* множества M в группе G называется множество $C_G(M) = \{g \mid gm = mg, g \in M, m \in M\}$.

Пример 2.2.6.

1. Нормализатор и централизатор множества M в группе G являются подгруппами группы G.

2. В абелевых группах централизатор любого элемента совпадает со всей группой.

3. В группе подстановок третьей степени централизаторы всех элементов совпадают с циклическими группами, порожденными этими элементами.

Теорема 2.2.3. Если M — непустое подмножество конечной группы G, то число подмножеств, сопряженных с M, совпадает с индексом $|G : N_G(M)|$. В частности, $|a^G| = |G : N_G(a)|$.

Строение группы во многом определяется перестановочностью ее элементов.

Определение 2.2.7. *Центром* группы G называется множество $Z(G) = C_G(G)$.

Пример 2.2.7.

1. Абелевая группа совпадает со своим центром.

2. В группе подстановок третьей степени центр группы является единичным.

Если H — нормальная подгруппа группы G, то правые и левые смежные классы по ней совпадают, поэтому просто говорим о множестве G/H смежных классов по подгруппе H. Легко видеть, что $aHbH = abH$, т.е. множество G/H замкнуто относительно поэлементного умножения классов.

Определение 2.2.8. Если $H \lhd G$, то множество смежных классов группы G по подгруппе H образует группу, которая называется *фактор-группой* группы G по подгруппе H.

Теорема 2.2.4. Порядок фактор-группы G/H равен индексу нормальной подгруппы H, т.е. $|G/H| = |G : H|$.

Пример 2.2.8. Найдите все фактор-группы группы S_3.

Среди подгрупп группы S_3 со своими сопряженными совпадают подгруппы $E = H_1$, $H = H_5$ и $S_3 = H_6$. Поэтому по теореме 2.2.2 они нормальны в S_3.

E — единичная подгруппа, поэтому

$$S_3/E = \{E, (12)E, (13)E, (23)E, (123)E, (132)E\} \simeq S_3.$$

Т.к. $S_3/H_6 = S_3/S_3 = \{S_3\}$, то S_3/H_6 — группа, изоморфная единичной группе.

Осталось рассмотреть нормальную подгруппу $H = H_5$. Ее порядок равен 3, а порядок S_3/H равен 2 по теореме 2.2.4. Поэтому S_3/H — циклическая группа порядка 2 (см. следствие 2.2.3). Смежные классы S_3 по H исчерпываются двумя классами: H и $(12)H$.

Таким образом, группа S_3 имеет три фактор-группы: $S_3/E \cong S_3$, $S_3/S_3 \cong E$ и $S_3/H = \{H, (12)H\}$, где E — единичная подгруппа, $H = \langle(123)\rangle = \{e, (123), (132)\}$.

Реализация в системе GAP

В системе GAP работу со смежными классами можно проводить, используя следующие функции:

- RightCoset(U, g) возвращает правый смежный класс с представителем g;
- RightCosets(G, U) возвращает все правые смежные классы группы G по подгруппе H;
- Representative(Ug) возвращает представителя смежного Ug;

• CosetDecomposition(G ,U) возвращает разложение группы G в правые смежные классы по подгруппе U.

Следует обратить внимание на то, что в GAP умножение подстановок выполняется слева направо, а не справа налево, как это принято в алгебре. Это связано с тем, что для образа точки i под действием подстановки p можно использовать как обозначение $p(i)$, так и обозначение $i\hat{}p$. В GAP принят за основу второй вариант записи (поскольку запись $p(i)$ интерпретировалась бы как обращение к функции p с аргументом i). Тогда выполняется соотношение $i\hat{}(p1 * p2) = (i\hat{}p1)\hat{}p2$, соответствующее правилу $(p1*p2)(i) = p1(p2(i))$. Поэтому при использовании функций RightCoset(U, g), RightCosets(G, U), CosetDecomposition(G ,U) для групп подстановок мы будем находить левые смежные классы и левое разложение по подгруппе.

Пример 2.2.9. Из каких элементов состоит левый смежный класс симметрической группы S_3 по подгруппе $\langle(12)\rangle$ с представителем (123).

```
gap> H:=Group((1,2));
Group([ (1,2) ])
gap> c:=RightCoset(H,(1,2,3));
RightCoset(Group( [ (1,2) ] ),(1,2,3))
gap> Representative(c);
(1,2,3)
```

```
gap> Size(c);
2
gap> AsList(c);
[ (1,2,3), (1,3) ]
```

Пример 2.2.10. Из каких элементов состоит правый смежный класс группы $GL(2,\mathbb{R})$ по подгруппе $H = \left\langle \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \right\rangle$ с представителем $g = \begin{bmatrix} 3 & 1 \\ 1 & -2 \end{bmatrix}$.

```
gap> A:=[[1,0],[0,-1]];
[ [ 1,  0 ], [ 0, -1 ] ]
gap> L:=Group(A);
Group([ [ [ 1,  0 ], [ 0, -1 ] ] ])
gap> B:=[[3,1],[1,-2]];
[ [ 3,  1 ], [ 1, -2 ] ]
gap> RightCoset(L,B);
RightCoset(Group([ [ [ 1,  0 ], [ 0, -1 ] ] ]),[ [ 3,  1 ], [ 1, -2 ] ])
gap> AsList(last);
[ [ [ 3,  1 ], [ 1, -2 ] ], [ [ 3,  1 ], [ -1,  2 ] ] ]
```

В системе GAP левые смежные классы группы по подгруппе не представлены. Однако левый смежный класс gU может быть задан как мно-

жество обратных к элементам правого смежного класса Ug^{-1}.

Пример 2.2.11. Найдите разложение симметрической группы S_3 в левые смежные классы по подгруппе $\langle(12)\rangle$.

```
gap> G:=SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> H:=Group((1,2));
Group([ (1,2) ])
gap> CosetDecomposition(G,H);
[ [ (), (1,2) ], [ (1,3), (1,2,3) ], [ (1,3,2), (2,3) ] ]
```

В системе GAP работу с классами сопряженных элементов можно проводить, используя следующие функции:

- ConjugacyClasses(G) возвращает классы сопряженных элементов группы G;
- NrConjugacyClasses(G) возвращает количество классов сопряженных элементов;
- IsConjugate(G,x,y) определяет, являются ли элементы x и y сопряженными в группе G.

Строение группы можно изучить по ее подгруппам. Выяснить, является ли подмножество H группы G ее подгруппой можно, используя функцию:

- IsSubgroup(G,H): возвращает «true», если H — подгруппа группы G, и «false» — в противном случае.

Чтобы найти все подгруппы группы G можно воспользоваться функцией:

- ConjugacyClassesSubgroups(G): возвращает список классов сопряженных подгрупп группы G.

Пример 2.2.12. Найдите все подгруппы группы S_4.

```
gap> G:=SymmetricGroup(4);
Sym( [ 1 .. 4 ] )
gap> c:=ConjugacyClassesSubgroups(G);
[ Group( () )^ G, Group( [ (1,3)(2,4) ] )^ G, Group( [ (3,4) ] )^ G,
Group( [ (2,4,3) ] )^ G, Group( [ (1,4)(2,3), (1,3)(2,4) ] )^ G,
Group( [ (1,2)(3,4), (3,4) ] )^ G, Group( [ (1,2)(3,4), (1,3,2,4) ] )^
G,  Group( [ (3,4), (2,4,3) ] )^ G, Group( [ (1,3)(2,4), (1,4)(2,3),
(1,2) ] )^ G,  Group( [ (1,3)(2,4), (1,4)(2,3), (2,4,3) ] )^ G,
Group( [ (1,3)(2,4), (1,4)(2,3), (2,4,3), (1,2) ] )^ G ]
gap> Size(c);
11
gap> m:=[];
[]
gap> for i in [1..Size(c)] do
> cc:=AsList(c[i]);
> for j in [1..Size(cc)] do
```

```
> Add(m,cc[j]);
> od;
> od;
gap> cc;
[ Group([ (1,3)(2,4), (1,4)(2,3), (2,4,3), (1,2) ]) ]
gap> m;
[ Group(()), Group([ (1,3)(2,4) ]), Group([ (1,4)(2,3) ]), Group([
(1,2)(3,4) ]), Group([ (3,4) ]), Group([ (2,4) ]), Group([ (2,3) ]),
Group([ (1,4) ]), Group([ (1,3) ]), Group([ (1,2) ]), Group([
(2,4,3) ]), Group([ (1,3,2) ]), Group([ (1,3,4) ]), Group([ (1,4,2)
]), Group([ (1,4)(2,3), (1,3)(2,4) ]), Group([ (1,2)(3,4), (3,4) ]),
Group([ (1,3)(2,4), (2,4) ]), Group([ (1,4)(2,3), (2,3) ]), Group([
(1,2)(3,4), (1,3,2,4) ]), Group([ (1,3)(2,4), (1,2,3,4) ]), Group([
(1,4)(2,3), (1,2,4,3) ]), Group([ (3,4), (2,4,3) ]), Group([ (1,3),
(1,3,2) ]), Group([ (1,3), (1,3,4) ]), Group([ (1,4), (1,4,2) ]),
Group([ (1,3)(2,4), (1,4)(2,3), (1,2) ]), Group([ (1,2)(3,4),
(1,4)(2,3), (1,3) ]), Group([ (1,2)(3,4), (1,3)(2,4), (1,4) ]),
Group([ (1,3)(2,4), (1,4)(2,3), (2,4,3) ]), Group([ (1,3)(2,4),
(1,4)(2,3), (2,4,3), (1,2) ]) ]
gap> Size(m);
30
```

Как видно, количество подгрупп в симметрической группе S_4 рав-

но 30.

Пример 2.2.13. Найдите все фактор-группы группы S_3.

```
gap> G:=SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> n:=NormalSubgroups(G);
[ Sym( [ 1 .. 3 ] ), Group([ (1,2,3) ]), Group(()) ]
gap> f:=List(n,i->FactorGroup(G,i));
[ Group(()), Group([ f1 ]), Sym( [ 1 .. 3 ] ) ]
gap> List(f,StructureDescription);
[ "1 "C2 "S3"]
```

2.3 Произведения групп. Прямое произведение. Гомоморфизм групп. Полупрямое произведение

Пусть G — мультипликативная группа, A и B — ее подгруппы. Напомним, что произведение $AB = \{ab \mid a \in A, b \in B\}$. Произведение AB является подмножеством в G. Если $AB = G$, то говорят, что группа G является произведением своих подгрупп A и B. В этом случае каждый элемент $g \in G$ представим в виде $g = ab$, где $a \in A$, $b \in B$.

Определение 2.3.1. Произведение $G = AB$ называется *прямым*, если подгруппы A и B нормальны в G и $A \cap B = 1$.

Прямое произведение в этом случае записывают так: $G = A \times B$.

Теорема 2.3.1. Пусть группа G является прямым произведением своих подгрупп A и B. Тогда:

а) каждый элемент $g \in G$ единственным образом представим в виде произведения $g = ab$, где $a \in A$, $b \in B$;

б) каждый элемент из A перестановочен с каждым элементом из B.

Обратно: если выполняются требования а и б, то $A \cap B = 1$, подгруппы A и B нормальны в G и $G = A \times B$.

Теорема 2.3.1 позволяет дать следующее определение прямого произведения, эквивалентное определению 2.3.1.

Определение 2.3.2. Группа G является *прямым произведением* своих подгрупп A и B, если:

а) каждый элемент $g \in G$ единственным образом представим в виде произведения $g = ab$, где $a \in A$, $b \in B$;

б) каждый элемент из A перестановочен с каждым элементом из B.

Определение прямого произведения мы дали для двух подгрупп. Для большего числа сомножителей определение прямого произведения выглядит так:

Определение 2.3.3. Группа G является *прямым произведением* своих подгрупп $A_1, A_2, \ldots, A_n$, если выполняются следующие :

1) все подгруппы A_i нормальны в G;

2) $A_i \cap A_1 \cdot \ldots \cdot A_{i-1} A_{i+1} \cdot \ldots \cdot A_n = 1$ для всех i;

3) $G = A_1 A_2 \cdot \ldots \cdot A_n$.

В этом случае пишут $G = A_1 \times A_2 \times \ldots \times A_n$.

Это определение можно заменить следующим, ему эквивалентным. Группа G является прямым произведением своих подгрупп $A_1, A_2, \ldots, A_n$, если:

а) каждый элемент g единственным образом представим в виде $g = a_1 a_2 \cdot \ldots \cdot a_n$, где $a_i \in A_i, i = 1, 2, \ldots, n$;

б) элементы из любых двух подгрупп A_i и A_j, $i \neq j$ перестановочны между собой.

Определение 2.3.4. *Примарной* называется группа, порядок которой есть степень некоторого простого числа. Группа, которая не может быть разложена в прямое произведение двух своих собственных подгрупп, называется *неразложимой*.

Пример 2.3.1.

1. Очевидно, неразложимыми будут все простые группы.

2. Симметрическая группа S_3 степени 3 содержит нетривиальную нормальную подгруппу $\langle (123) \rangle$. Поэтому группа S_3 неразложима.

Теорема 2.3.2. Циклическая примарная группа является неразложимой группой.

Следующая теорема показывает, что каждая циклическая группа составного порядка разложима в прямое произведение своих подгрупп.

Теорема 2.3.3. Если $\langle g \rangle$ — циклическая группа порядка nm, где n, m — взаимно простые числа, то $\langle g \rangle = \langle g^m \rangle \times \langle g^n \rangle$.

Пример 2.3.2. Пусть $G = \langle g \rangle$ — циклическая группа порядка 100.

Т.к. $100 = 2^2 5^2$, то по теореме 2.3.3 группа

$$\langle g \rangle = \langle g^{5^2} \rangle \times \langle g^{2^2} \rangle$$

является прямым произведением своих примарных подгрупп: $\langle g^{5^2} \rangle$ порядка 2^2 и $\langle g^{2^2} \rangle$ порядка 5^2. По теореме 2.3.2 подгруппы $\langle g^{25} \rangle$ и $\langle g^4 \rangle$ неразложимы.

Пример 2.3.3. Пусть группа $G = \langle (1254)(367) \rangle$. Разложима ли G в прямое произведение своих подгрупп?

Легко показать, что $|G| = 12$. Т.к. $12 = 2^2 3$, то по теореме 2.3.3 G представима в виде прямого произведения своих подгрупп:

$$G_1 = \langle (1254)(367)^4 \rangle, G_2 = \langle (1254)(367)^3 \rangle.$$

$$(1254)(367)^2 = (15)(24)(376),$$

$$(1254)(367)^3 = (1452),$$

$$(1254)(367)^4 = (367).$$

Таким образом, $G = \langle (1452) \rangle \times \langle (367) \rangle$.

Пример 2.3.4. Пусть $A = \langle (1576) \rangle, B = \langle (4328) \rangle$ — подгруппы симметрической группы S_8 степени 8. Существует ли в S_8 подгруппа $H = A \times B$?

Легко видеть, что

$$A = \{e, (1576), (17)(56), (1675)\},$$

$$B = \{e, (4328), (42)(38), (4823)\},$$
$$A \cap B = 1.$$

Покажем, что $AB = BA$.

$$Ae = eA,$$

$$A(4328) = \{(4328), (1576)(4328), (17)(56)(4328), (1675)(4328)\} = (4328)A,$$
$$A(48)(38) = \{(42)(38), (1576)(42)(38), (17)(56)(42)(38), (1675)(42)(38)\} =$$
$$= (42)(38)A,$$
$$A(4823) = \{(4823), (1576)(4823), (17)(56)(4823), (1675)(4823)\} = (4823)A.$$

Т.к. $Ab = bA$ для всех $b \in B$, то $AB = BA$. По теореме 2.1.2 произведение AB — подгруппа. Нетрудно заметить, что $A \lhd AB$. Аналогично можно показать, что $Ba = aB$ для всех $a \in A$, т.е. $B \lhd AB$.

Таким образом, в S_8 существует подгруппа $AB = A \times B$.

Определение 2.3.5. Отображение ϕ мультипликативной группы G в мультипликативную группу S называется *гомоморфным*, или гомоморфизмом, если $\phi(ab) = \phi(a)\phi(b)$ для любых a, b из G. В частности, если ϕ — биекция, то ϕ называется *изоморфизмом* и обозначается $G \cong S$.

Определение 2.3.6. Множество всех элементов из G, которые при гомоморфизме ϕ отображаются в единицу мультипликативной группы S, называется *ядром* гомоморфизма ϕ и обозначается $\mathrm{Ker}\ \phi$.

Пример 2.3.5.

1. Отображение ϕ группы целых чисел $\mathbb{Z}$ по сложению на аддитивную группу кольца $\mathbb{Z}_n$ классов вычетов по модулю n — гомоморфизм.

2. Отображение ϕ симметрической группы S_n подстановок степени n на мультипликативную группу кольца $\mathbb{Z}_n$ — гомоморфизм.

Теорема 2.3.4. Ядро любого гомоморфизма ϕ группы G является нормальной подгруппой группы G.

Определение 2.3.7. Пусть H — нормальная подгруппа группы G. Поставим каждому элементу x группы G соответствующий смежный класс xH и получим отображение группы G на фактор-группу G/H. Это отображение будет гомоморфизмом: $\phi(ab) = (ab)H = aH \cdot bH = \phi(a) \cdot \phi(b)$. Полученный гомоморфизм называется *естественным гомоморфизмом* группы G на фактор-группу G/H. Таким образом, нормальные подгруппы, и только они, являются ядрами гомоморфизмов.

Теорема 2.3.5. Пусть дан гомоморфизм ϕ группы G на группу S и H — ядро этого гомоморфизма. Тогда группа S изоморфна фактор-группе G/H, причем гомоморфизм ϕ равен последовательному выполнению естественного гомоморфизма $\varepsilon : G \to G/H$ и изоморфизма $\tau : G/H \to S$.

Теорема 2.3.6. Пусть H и A — нормальные подгруппы группы G и H — подгруппа группы A. Тогда фактор-группа $(G/H)/(A/H)$ изоморфна фактор-группе G/A.

Пример 2.3.6.

1. Группа положительных действительных чисел $\mathbb{R}^+$ по умножению изоморфна группе действительных чисел $\mathbb{R}$ по сложению. Изоморфное отображение получается, если всякому положительному действительному числу поставим в соответствие его логарифм по основанию 10. Равенство $lg(ab) = lg(a) + lg(b)$ показывает, что это отображение является изоморфным.

2. Группа корней n-й степени из единицы по умножению изоморфна аддитивной группе кольца $\mathbb{Z}_n$ классов вычетов по модулю n.

3. Множество четных чисел можно взаимнооднозначно отобразить на множество чисел, кратных числу 3, если всякому четному числу вида $2k$ поставить в соответствие число вида $3k$, лежащее во втором .

Всякое множество с операцией изоморфно, очевидно, самому себе: для этого достаточно взять тождественное отображение множества на себя. Следовательно, отношение изоморфизма является рефлексивным. Легко видеть, что оно также является симметричным (из $M_1 \cong M_2$ следует $M_2 \cong M_1$) и транзитивным (из $M_1 \cong M_2$ и $M_2 \cong M_1$ следует $M_1 \cong M_3$). Выполнение трех этих свойств означает, что изоморфизм является отношением эквивалентности на множестве групп. Из определения изоморфизма следует, что изоморфные множества имеют одинаковую мощность, в частности, если они конечны, то состоят из одинакового числа элементов.

Изоморфные группы отличаются друг от друга природой своих эле-

ментов и, быть может, названием операций. Они неразличимы с точки зрения свойств операций. Все, что может быть доказано для некоторого множества с операцией на основании свойств этой операции, но без использования конкретной природы элементов множества, автоматически переносится на все множества с операцией, изоморфные данному. Тем самым алгебраическая операция выделяется в качестве истинного объекта изучения.

Пример 2.3.7. Покажите, что факор-группа $GL(n,\mathbb{P})/SL(n,\mathbb{P})$ изоморфна мультипликативной группе $\mathbb{P}^\sharp$ поля $\mathbb{P}$.

Определим отображение $det : GL(n,\mathbb{P}) \mapsto \mathbb{P}^\sharp$, которое каждой матрице A из полной линейной группы $GL(n,\mathbb{P})$ степени n над полем $\mathbb{P}$ ставит в соответствие ее определитель. Т.к. определитель произведения двух матриц равен произведению определителей, т.е $det\,(AB) = detA \cdot detB$, то отображение det – гомоморфизм. Каждый элемент $a \in \mathbb{P}^\sharp$ будет определителем матрицы

$$A = \begin{pmatrix} a & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots\dots\dots & \dots\dots\dots & \dots\dots\dots & \\ 0 & 0 & \dots & 1 \end{pmatrix}, \text{ поэтому } \mathrm{Im}\ det = \mathbb{P}^\sharp.$$ Ядро det состоит из матриц с единичным определителем, поэтому $\mathrm{Ker}\ det = SL(n,\mathbb{P})$ — специальная линейная группа, а по теореме 2.3.4 подгруппа $SL(n,\mathbb{P})$ нормальна в $GL(n,\mathbb{P})$. По теореме 2.3.5 фактор-группа $GL(n,\mathbb{P})/SL(n,\mathbb{P})$ изоморфна мультипликативной группе $\mathbb{P}^\sharp$ поля $\mathbb{P}$.

Пример 2.3.8. Пусть $G = S_n$ — симметрическая группа степени n, а $H = \{-1; 1\}$ — мультипликативная группа. Докажите, что отображение $sgn : \tau \mapsto sgn\tau$ является гомоморфизмом. Найдите ядро и образ гомоморфизма sgn.

По условию, четным подстановкам ставится в соответствие 1, а нечетным — (-1). Поскольку знак произведения перестановок равен знаков, т.е. $sgn\,(\tau\sigma) = sgn\,\tau \cdot sgn\,\sigma$, то sgn — гомоморфизм. Ядро $\mathrm{Ker}\ sgn$ состоит из четных подстановок, поэтому $\mathrm{Ker}\ sgn = A_n$ — знакопеременная группа. Легко видеть, что sgn – сюръекция, поэтому $\mathrm{Im}\ sgn = H$.

Определение 2.3.8. Пусть A и B — группы и φ — гомоморфизм A в $\mathrm{Aut}B$. Тогда существует группа G со следующими свойствами: $G = AB$, $B \lhd G$ и $A \cap B = 1$. Эту группу называют *полупрямым произведением* групп A и B относительно φ и обозначают через $G = A[B]$.

Пример 2.3.9.

1. Симметрическая группа S_n раскладывается в полупрямое произведение $S_n = [A_n]\langle(12)\rangle$, т.к. A_n нормальна в S_n, $\langle(12)\rangle = \{e, (12)\}$ — подгруппа из двух элементов и $S_n = A_n\langle(12)\rangle$.

2. Полная линейная группа $GL(n, F)$ является полупрямым произведением: $GL(n, F) = [SL(n, F)]\{\mathrm{diag}(\lambda, 1, \ldots, 1), \lambda \in F\backslash\{0\}\}$.

Реализация в системе GAP

Функция

• DirectProduct(G,H) возвращает прямое произведение групп, заданных в качестве аргументов.

GAP будет стараться выбрать оптимальный вид группы, которая является прямым произведением других групп. Например, прямое произведение групп подстановок вновь будет группой подстановок.

```
gap> g:=Group((1,2,3),(1,2));;
gap> d:=DirectProduct(g,g,g);
Group( [ (1,2,3), (1,2), (4,5,6), (4,5), (7,8,9), (7,8) ] )
gap> Size(d);
216
gap> IsPermGroup(d);
true
```

Пример 2.3.10. Пусть группа $G = \langle (1254)(367) \rangle$. Разложима ли G в прямое произведение своих подгрупп?

```
gap> G:=Group((1,2,5,4)(3,6,7));
Group([ (1,2,5,4)(3,6,7) ])
gap> N:=NormalSubgroups(G);
[ C12, Group([ (1,5)(2,4), (3,6,7) ]), Group([ (1,2,5,4), (1,5)(2,4)
]), Group([ (1,5)(2,4) ]), Group([ (3,6,7) ]), Group(()) ]
```

```
gap> List(N,Size);
[ 12, 6, 4, 2, 3, 1 ]
gap> Intersection(N[3],N[5]);
Group(())
gap> D:=DirectProduct(N[3],N[5]);
Group([ (1,2,4,3), (1,4)(2,3), (5,6,7) ])
gap> D=G;
false
gap> Size(G);
12
gap> Size(D);
12
gap> IsomorphismGroups(G,D);
[ (3,6,7), (1,4,5,2) ] -> [ (5,6,7), (1,2,4,3) ]
```

Пример 2.3.11. Пусть $A = \langle(1576)\rangle, B = \langle(4322)\rangle$ — подгруппы симметрической группы S_8 степени 8. Существует ли в S_8 подгруппа $H = A \times B$?

```
gap> A:=Group((1,5,7,6));
Group([ (1,5,7,6) ])
gap> B:=Group((4,3,2,8));
Group([ (2,8,4,3) ])
```

```
gap> D:=DirectProduct(A,B);
Group([ (1,2,4,3), (5,8,7,6) ])
gap> G:=SymmetricGroup(8);
Sym( [ 1 .. 8 ] )
gap> IsSubgroup(G,D);
true
```

Полупрямое произведение группы N и группы G задается при помощи функции

• SemidirectProduct(G, alpha, N): возвращает полупрямое произведение группы N на группу G, действующую на N через *alpha*, где *alpha* — гомоморфизмом группы G в группу автоморфизмов группы N.

В GAP существует еще один вариант задания полупрямого произведения:

• SemidirectProduct(autgp,N) — упрощенная запись функции SemidirectProduct(autgp,IdentityMapping(autgp),N). Здесь *autgp* должна являться группой автоморфизмов группы N. При этом, если *autgrp* не была получена с помощью операции AutomorphismGroup, рекомендуется предварительно проверить, что *autgrp* действительно состоит из групповых автоморфизмов, с помощью функции IsGroupOfAutomorphisms(autgrp).

Пример 2.3.12. Постройте полупрямое произведение $[E_{3^2}]S_3$. Здесь E_{3^2} — элементарная абелевая подгруппа порядка 9.

```
gap> N:=ElementaryAbelianGroup(9);
<pc group of size 9 with 2 generators>
gap> G:=SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> autN:=AutomorphismGroup(N);
<group of size 48 with 3 generators>
gap> c:=ConjugacyClassesSubgroups(autN);;
gap> Size(c);
16
gap> c:=List(c,Representative);
[ <trivial group>, <group of size 2 with 1 generators>, <group of
size 2 with 1 generators>, <group of size 3 with 1 generators>,
<group of size 4 with 2 generators>, <group of size 4 with 2
generators>, <group of size 6 with 2 generators>, <group of size
6 with 2 generators>, <group of size 6 with 2 generators>, <group
of size 8 with 3 generators>, <group of size 8 with 3 generators>,
<group of size 8 with 3 generators>, <group of size 12 with 3
generators>, <group of size 16 with 4 generators>, <group of size
24 with 4 generators>, <group of size 48 with 5 generators> ]
```

```
gap> f:=Filtered(c,i->Size(i)=Size(G));
[ <group of size 6 with 2 generators>, <group of size 6 with 2
generators>, <group of size 6 with 2 generators> ]
gap> autgr1:=f[1];
<group of size 6 with 2 generators>
gap> StructureDescription(last);
"C6"
gap> autgr2:=f[2];
<group of size 6 with 2 generators>
gap> StructureDescription(last);
"S3"
gap> autgr3:=f[3];
<group of size 6 with 2 generators>
gap> StructureDescription(last);
"S3"
gap> S:=SemidirectProduct(autgr2,N);
<pc group with 4 generators>
gap> Size(S);
54
gap> StructureDescription(S);
"((C3 x C3) : C3) : C2"
```

2.4 Классы групп. Группы малых порядков. Инварианты разрешимых групп

Определение 2.4.1. Ряд подгрупп $G = H_0 \supseteq H_1 \supseteq \ldots \supseteq H_t = 1$ называется:

1) *субнормальным*, если $H_{i+1} \lhd H_i$ для всех $i = 0, 1, \ldots, t-1$;

2) *нормальным*, если $H_i \lhd G$ для всех $i = 1, \ldots, t-1$;

3) *главным*, если ряд является нормальным и $H_i/H_{i+1} \cdot \lhd\, G/H_{i+1}$ для всех $i = 0, 1, \ldots, t-1$. В частности, H_i/H_{i+1} называется *главным фактором*;

4) *композиционным*, если ряд является субнормальным и фактор-группа H_i/H_{i+1} — простая группа. В частности, H_i/H_{i+1} называется *композиционным фактором*.

Определение 2.4.2.

Группу G называют:

1) *примарной*, если $|\pi(G)| = 1$;

2) *бипримарной*, если $|\pi(G)| = 2$;

3) *нильпотентной*, если все силовские подгруппы в группе G ;

4) *p-разрешимой*, если существует нормальный ряд, факторы которого либо p-группы, либо p'-группы;

s17 5) *разрешимой*, если существует такое натуральное n, что $G^{(n)} = 1$. Разрешимая группа является p-разрешимой для всех $p \in \pi(G)$. Наименьшее натуральное n, для которого $G^{(n)} = 1$, называется *производной*

длиной группы G и обозначается через $d(G)$. Другими словами, под производной длиной разрешимой группы понимают наименьшую из длин её нормальных рядов с абелевыми факторами;

6) *сверхразрешимой*, если она имеет нормальный ряд с циклическими факторами;

7) *метанильпотентной*, если она содержит нильпотентную нормальную подгруппу, фактор-группа по которой нильпотентна;

8) *метабелевой*, если она содержит абелеву нормальную подгруппу, фактор-группа по которой абелева.

Пример 2.4.1.

1. Прозводная длина метабелевой группы не превышает 2.

2. Производная длина симметрической группы S_3 равна 2, т.к. $S_3 = [Z_3]Z_2$.

3. Производная длина симметрической группы S_4 равна 3, т.к. для группы S_4 существует следующий нормальный ряд наименьшей длины с абелевыми факторами $1 \leq E_4 \leq A_4 \leq S_4$. Здесь E_4 — элементарная абелевая подгруппа порядка 4.

Определение 2.4.3. *Дисперсивная группа* — группа, обладающая нормальным рядом, факторы которого изоморфны силовским подгруппам. Более точно, пусть ϕ — некоторое упорядочение простых чисел. Запись $p\phi q$ означает, что p предшествует q в упорядочении ϕ, $p \neq q$. Группа G порядка $p_1^{\alpha_1}p_2^{\alpha_2}\dots p_n^{\alpha_n}$ называется *ϕ-дисперсивной*, если $p_1\phi p_2\phi\dots\phi p_n$ и для любого i группа G имеет нормальную подгруппу порядка $p_1^{\alpha_1}p_2^{\alpha_2}\dots p_i^{\alpha_i}$,

т.е. группа G имеет нормальный ряд

$$1 \subset G_1 \subset G_2 \subset \ldots \subset G_{n-1} \subset G, \qquad (2.4.1)$$

где $|G_1| = p_1^{\alpha_1}$, $|G_2| = p_1^{\alpha_1} p_2^{\alpha_2}$, $\ldots$, $|G_i| = p_1^{\alpha_1} p_2^{\alpha_2} \cdot \ldots \cdot p_i^{\alpha_i}, \ldots$, $|G_{n-1}| = p_1^{\alpha_1} p_2^{\alpha_2} \cdot \ldots \cdot p_{n-1}^{\alpha_{n-1}}$. В этом случае у ряда (3.4.1) факторы изоморфны силовским подгруппам $G_1/1 \simeq G_{p_1}$, $G_2/G_1 \simeq G_{p_2}$, $\ldots$, $G_n/G_{n-1} \simeq G_{p_n}$. Если при этом упорядочение ϕ таково, что $p \phi q$ всегда влечет $p > q$, то ϕ-дисперсивная группа называется *дисперсивной по Оре*. Дисперсивной группой называют группу, являющуюся ϕ-дисперсивной для некоторого упорядочениz ϕ.

Определение 2.4.4. *Подгруппой Фиттинга* группы G называется подгруппа $F(G)$ группы G, являющаяся произведением всех нильпотентных нормальных в G подгрупп.

Определение 2.4.5. Пусть G — группа и пусть $F_0(G) = 1$,
$F_1(G) = F(G)$ — подгруппа Фиттинга группы G,
$F_2(G)/F_1(G) = F(G/F_1(G)), \ldots, F_i(G)/F_{i-1}(G) = F(G/F_{i-1}(G)), \ldots$.
Ясно, что $1 = F_0(G) \subseteq F_1(G) \subseteq F_2(G) \subseteq \ldots$
В разрешимой неединичной группе подгруппа Фиттинга отлична от

Пример 2.4.2.

1. Нильпотентная длина группы S_3 равна 2.
2. Нильпотентная длина группы S_4 равна 3.
3. Нильпотентная длина группы A_4 равна 2.

Определение 2.4.6. Для p-разрешимой группы можно определить (p', p)-ряд:

$$1 = P_0 \subseteq N_0 \subseteq P_1 \subseteq N_1 \subseteq P_2 \subseteq \ldots \subseteq P_l \subseteq N_l = G,$$

где $N_i/P_i = O_{p'}(G/P_i)$ — наибольшая нормальная p'-подгруппа в G/P_i, а $P_{i+1}/N_i = O_p(G/N_i)$ — наибольшая нормальная p-подгруппа в G/N_i. Наименьшее натуральное число l такое, что $N_l = G$, называют *p-длиной* группы G и обозначают через $l_p(G)$.

Пример 2.4.3.

1. p-длина метанильпотентной группы не превышает 1 для произвольного простого числа p.

2. 2-длина и 3-длина симметрической группы S_3 равна 1, т.к. $S_3 = [Z_3]Z_2$.

3. 2-длина симметрической группы S_4 равна 2, а 3-длина равна 1, т.к. для группы S_4 существует следующий нормальный ряд $1 \leq E_4 \leq A_4 \leq S_4$ с 2-факторами и 3-факторами.

Реализация в системе GAP

Библиотека групп малых порядков в системе GAP содержит все конечные группы, порядок которых не превышает 2 000, за исключением групп порядка 1 024. Каждая группа из библиотеки имеет свой номер, обозначающий ее тип изоморфизма. Этот номер имеет вид [n1, n2], где n1 — порядок группы, n2 — ее номер в каталоге групп порядка n1. Группу, имеющую тип изоморфизма [n1, n2], можно вызвать из библиотеки с помощью функции SmallGroup, например:

```
S:=SmallGroup(24,12);
<pc group of size 24 with 4 generators>
```

С другой стороны, для многих групп возможно определение их типа изоморфизма с помощью функции IdGroup. Так, например, симметрическая группа степени 4 может быть получена, как группа с номером 12 из библиотеки групп порядка 24:

```
S:=SymmetricGroup(4);
Sym( [ 1 .. 4 ] )
gap> IdGroup(S);
[ 24, 12 ]
```

Для отбора групп из библиотеки используется функция AllSmallGroups в комбинации с различными аргументами, первым из которых всегда является Size, вторым — порядок требуемых групп). Далее могут быть

записаны другие пары аргументов, где в каждой паре первый аргумент — функция для отбора групп, второй — ее требуемое значение. В следующем примере получается список всех неабелевых групп порядка 24:

```
gap> l:=AllSmallGroups(Size,24, IsAbelian, false);;
gap> Length(l);
12
```

Таким образом, существует 12 таких групп.

Принадлежность к классу абелевых, нильпотентных, разрешимых, сверхразрешимых и др. групп в системе GAP можно проверить при помощи следующих функций:

- IsAbelian(G) возвращает «true», если группа G является абелевой (коммутативной) и «false» — в противном случае;
- IsCyclic(G) возвращает «true», если группа G является циклической, и «false» – в противном случае;
- IsAlternatingGroup(G) возвращает «true», если группа G является знакопеременной, и «false» – в противном случае;
- IsElementaryAbelian(G) возвращает «true», если группа G является элементарной абелевой, и «false» – в противном случае;
- IsNilpotent(G) возвращает «true», если группа G является нильпотентной, и «false» — в противном случае;
- IsSolvable(G) возвращает «true», если группа G является разрешимой, и «false» – в противном случае;

• IsSupersolvableGroup(G) возвращает «true», если группа G является сверхразрешимой, и «false» – в противном случае;

• IsSimpleGroup(G) возвращает «true», если группа G является простой, и «false» – в противном случае.

Пример 2.4.4. Исследуйте силовскую 2-подгруппу симметрической группы S_3.

```
gap> G:=SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> S2:=SylowSubgroup(G,2);
Group([ (1,2) ])
gap> IsTrivial(S2);
false
gap> IsAbelian(S2);
true
gap> IsCyclic(S2);
true
gap> IsElementaryAbelian(S2);
true
gap> IsNilpotent(S2);
true
gap> IsSolvable(S2);
true
```

Изучить строение разрешимых групп можно путем нахождения оценок инвариантов (производной длины, нильпотентной длины, p-длины и других).

В системе GAP реализована функция для нахождения производной длины разрешимой группы:

• DerivedLength(G) возвращает значение производной длины группы G. Если группа не является разрешимой, то возвращает значение 0.

```
gap> G:=AlternatingGroup(5);
Alt( [ 1 .. 5 ] )
gap> DerivedLength(G);
0
gap> G1:=AlternatingGroup(4);
Alt( [ 1 .. 4 ] )
gap> DerivedLength(G1);
2
gap> G2:=SymmetricGroup(4);
Sym( [ 1 .. 4 ] )
gap> DerivedLength(G2);
3
gap> G3:=SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> DerivedLength(G3);  2
```

Встроенной функции, связанной с нахождением нильпотентной длины, в системе GAP нет. Поэтому возникает следующая задача.

Пример 2.4.5. Используя определение 2.4.5, напишите функцию, которая находит нильпотентную длину разрешимой группы.

Функция FittingSubgroup(G) возвращает подгруппу Фиттинга группы G.

```
NilpLength:=function(G)
local i,F;
if IsSolvable(G)<>true then return 0;
else
i:=0;
while IsNilpotent(G)<>true do
F:=FittingSubgroup(G);
G:=FactorGroup(G,F);
i:=i+1;
od;
if i=0 then return 1;
else return 1+i;
fi;
fi;
end;
```

```
gap> Read("C:/gap4r7/bin/NilpLength.g");
gap> G:=AlternatingGroup(5);
Alt( [ 1 .. 5 ] )
gap> NilpLength(G);
0
gap> G1:=SymmetricGroup(4);
Sym( [ 1 .. 4 ] )
gap> NilpLength(G1);
3
gap> G2:=SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> NilpLength(G2);
2
gap> G3:=AlternatingGroup(4);
Alt( [ 1 .. 4 ] )
gap> NilpLength(G3);
2
```

Встроенной функции, связанной с нахождением p-длины, в системе GAP нет.

Пример 2.4.6. Используя определение 2.4.6, напишите функцию, которая находит p-длину p-разрешимой группы.

Будем использовать функции:

• PCore(G,p): возвращает наибольшую нормальную p-подгруппу группы G;

• Core(G,U): возвращает наибольшую нормальную подгруппу группы G, содержащуюся в подгруппе U;

• HallSubgroup(G,pi): возвращает холлову π-подгруппу группы G.

Для написания основного алгоритма нам понадобится вспомогательная функция для нахождения наибольшей нормальной p'-подгруппы.

```
Op':=function(G,p)
local piG,pi,U,G1;
piG:=Set(PrimeDivisors(Size(G)));
pi:=Filtered(piG,m->m<>p);
U:=HallSubgroup(G,pi);
G1:=Core(G,U);
return G1;
end;
plength:=function(G,p,fun)
local N,P,i;
N:=fun(G,p);
i:=0;
while N<>G do
G:=FactorGroup(G,N);
P:=PCore(G,p);
```

```
G:=FactorGroup(G,P);
N:=fun(G,p);
i:=i+1;
od;
return i;
end;
gap> Read("C:/gap4r7/bin/Op'.g");;  gap>
Read("C:/gap4r7/bin/plength.g");;
gap> G:=SymmetricGroup(3);
Sym( [ 1 .. 3 ] )
gap> plength(G,3,Op');
1
gap> plength(G,2,Op');
1
gap> G1:=SymmetricGroup(4);
Sym( [ 1 .. 4 ] )
gap> plength(G1,2,Op');
2
gap> plength(G1,3,Op');
1
gap> G2:=AlternatingGroup(4);
Alt( [ 1 .. 4 ] )
```

```
gap> plength(G2,3,Op');
1
gap> plength(G2,2,Op');
1
```

РАЗДЕЛ 3
ЭЛЕМЕНТЫ ТЕОРИИ ЧИСЕЛ В СИСТЕМЕ GAP

3.1 Делимость целых чисел

Определение 3.1.1. *Целое число a делится на целое число b*, отличное от нуля, если существует целое число q такое, что верно равенство $a = bq$.

Введем символы, обозначающие «a делится на b»: $a{:}b$. Вместо выражения «a делится на b» говорят также «a кратно b», «b делитель a». Так же, как и в школьном курсе алгебры, числа a, b, q называем делимое, делитель, частное.

Лемма 3.1.1. Простейшие свойства делимости.

1. Нуль делится на любое отличное от нуля целое число a.
2. Любое целое число делится на 1, -1.
3. Любое целое число $a \neq 0$ делится само на себя.
4. Знак числа не влияет на делимость, т.е. если a делится на b, то a делится на $(-b)$, $(-a)$ делится на b, $(-a)$ делится на $(-b)$.
5. Если a делится на b и b делится на c, то a делится на c (транзитивность делимости).
6. Если каждое слагаемое суммы делится на некоторое целое число, то и сумма делится на это число. (Обратное утверждение неверно.)
7. Если одно из двух целых чисел делится на какое-либо целое число

b, то сумма делится на b тогда и только тогда, когда и второе число делится на b.

8. Если уменьшаемое и вычитаемое делятся на целое число b, то и их разность делится на это число. (Обратное утверждение неверно.)

9. Если хотя бы один из сомножителей делится на какое-либо целое число, то и произведение этих сомножителей делится на это число. (Обратное утверждение неверно.)

10. Если a делится на b и $a \neq 0$, то $|a| \geq |b|$.

Определение 3.1.2. *Целое число a делится с остатком на целое число b*, $b \neq 0$, если существуют целые числа $q,\ r$ такие, что $a = bq + r$, причем $0 \leq r < |b|$.

Теорема 3.1.1. (О делении с остатком). Для любых целых чисел a и b $(b \neq 0)$ существует единственная пара целых чисел $q,\ r$, удовлетворяющих условию $a = bq + r$, где $0 \leq r < |b|$.

Пример 3.1.1. Разделите ± 658 на ± 37.

Т.к. $629 = 37 \cdot 17 < 658 < 37 \cdot 18 = 666$, то $658 = 37 \cdot 17 + 29$. Здесь 17 — неполное частное, 29 — остаток.

Разделим -658 на 37. Т.к. $37 \cdot (-18) = -666 < -658 < 37 \cdot (-17) = -629$, то $-658 = 37 \cdot (-18) + 8$. Здесь -18 — неполное частное, 8 — остаток.

Разделим 658 на -37. Т.к. $629 = (-37) \cdot (-17) < 658 < (-37) \cdot (-18) = 666$, то $658 = -37 \cdot (-17) + 29$. Здесь -17 — неполное частное, 29 — оста-

ток.

Разделим -658 на -37. Т.к. $(-37)\cdot 18 = -666 < -658 < (-37)\cdot 17 = -629$, то $-658 = (-37)\cdot 18+8$. Здесь 18 — неполное частное, 8 — остаток.

Ответ: $658 = 37\cdot 17+29$, $-658 = 37\cdot(-18)+8$, $658 = -37\cdot(-17)+29$, $-658 = (-37)\cdot 18+8$.

Реализация в системе GAP

Деление целых чисел нацело и с остатком в системе компьютерной алгебры GAP реализуется при помощи следующих функций:

- a/b делит число a на число b (если a не делится нацело на b, то результатом будет рациональная дробь);
- DivisorsInt(n) возвращает список натуральных делителей целого числа n;
- a mod b возвращает остаток от деления a на b.

Пример 3.1.2. Вычислите: 144/12, 32/10.

```
gap> 144/12;
12
gap> 32/10;
16/5
```

Пример 3.1.3. Сколько чисел в интервале от 1 до 200 делится на 7?

```
gap> l := Filtered( [ 1 .. 200 ], i -> i mod 7 = 0);
[ 7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105, 112, 119,
126, 133, 140, 147, 154, 161, 168, 175, 182, 189, 196 ]
gap> Length(l);
28
```

Пример 3.1.4. Выясните является ли число 496 совершенным.

Напомним, что совершенным числом называется такое натуральное число a, у которого сумма всех натуральных делителей равна $2a$.

```
gap> Sum(DivisorsInt(496));
992
```

Пример 3.1.5. Разработайте функцию деления с остатком числа a на число b.

```
ostatok:=function(a,b)
local q,r;
r:=a mod b;
q:=(a-r)/b;
Print(a,"=",b,"*",q,"+",r);
return q;
end;
```

```
gap> Read("C:/gap4r7/bin/ostatok.g");
gap> ostatok(658,37);
658=37*17+29
17
gap> ostatok(-658,37);
-658=37*-18+8
-18
gap> ostatok(-658,-37);
-658=-37*18+8
18
gap> ostatok(658,-37);
658=-37*-17+29
-17
```

3.2 Наибольший общий делитель (НОД). Алгоритм Евклида. Наименьшее общее кратное (НОК)

Определение 3.2.1. *Общим делителем целых чисел* $a_1, a_2, \ldots, a_k$, $k \geq 2$ называется целое число, которое делит каждое из чисел a_i, $i = \overline{1,k}$.

Пусть среди чисел a_i хотя бы одно отлично от нуля. Тогда существует конечное число общих делителей, среди которых можно выбрать наибольший делитель (НОД). Заметим, что общим делителем любой со-

вокупности целых чисел является число 1. Поэтому наибольший общий делитель этих чисел будет равен либо 1, либо больше 1, т.е. НОД — число натуральное. Будем обозначать наибольший общий делитель целых чисел $a_1, a_2, \ldots, a_k$ через

$$\text{НОД}(a_1, a_2, \ldots, a_k).$$

Определение 3.2.2. Целые числа $a_1, a_2, \ldots, a_k$, $k \geq 2$, называются *взаимно простыми*, если их НОД равен 1.

Определение 3.2.3. Целые числа $a_1, a_2, \ldots, a_k$, $k \geq 2$, называются *попарно взаимно простыми*, если наибольший общий делитель любых двух чисел этой совокупности равен 1, т.е. $\text{НОД}(a_i, a_j) = 1$, где $i, j = \overline{1, k}$, $i \neq j$.

Теорема 3.2.1. (о линейном представлении наибольшего общего делителя целых чисел). Наибольший общий делитель d целых чисел $a_1, a_2, \ldots, a_k$, $k \geq 2$, представим в виде целочисленной линейной комбинации этих чисел, т.е. в форме $d = x_1a_1 + x_2a_2 + \ldots + x_ka_k$, где $x_i \in \mathbb{Z}$, $i = \overline{1, k}$.

Определение 3.2.4. *Алгоритмом Евклида* для двух целых чисел a и b, $b \neq 0$, называется процесс последовательного деления, который можно описать следующими равенствами с соответствующими условиями, выполняемыми для этих равенств:

$$a = bq_1 + r_1,\ 0 < r_1 < |b|,$$
$$b = r_1q_2 + r_2,\ 0 < r_2 < r_1,$$
$$r_1 = r_2q_3 + r_3,\ 0 < r_3 < r_2 \text{ и т.д.}$$

Вопрос о конечности данного процесса решается следующим образом: заметим, что остатки удовлетворяют условию $|b| > r_1 > r_2 > r_3 > \ldots$, т.е. образуют убывающий натуральный ряд, который убывать бесконечно не может, т.к. числа этого ряда натуральные. Следовательно, в этом процессе число остатков конечно, а значит, и сам процесс конечен. Этот процесс остановит нулевой остаток, т.к. следующий шаг алгоритма будет состоять в делении на нуль, что невозможно.

Пусть $r_{k+1} = 0$. Тогда предпоследний и последний шаги в алгоритме Евклида запишутся следующим образом

$$r_{k-2} = r_{k-1}q_k + r_k,\ 0 < r_k < r_{k-1},$$
$$r_{k-1} = r_k q_{k+1}.$$

Теорема 3.2.2. Наибольший общий делитель двух целых чисел равен последнему ненулевому остатку в алгоритме Евклида для этих чисел.

Пример 3.2.1. Вычислите НОД$(300, 85)$. Выразите НОД через исходные числа.

Составим алгоритм Евклида для чисел 300 и 85. Выполним последовательно деление с остатком: $300 = 85 \cdot 3 + 45$, $85 = 45 \cdot 1 + 40$, $45 = 40 \cdot 1 + 5$, $40 = 5 \cdot 8$. Последний отличный от нуля остаток в алгоритме Евклида является наибольшим общим делителем чисел 300 и 85, т.е. НОД$(300, 85) = 3$.

Выразим НОД$(300, 85)$ через исходные числа 300 и 85, двигаясь в алгоритме Евклида снизу вверх и последовательно выражая остатки:

НОД$(300, 85) = 5 = 45 - 40 = 45 - (85 - 45) = 2 \cdot 45 - 85 = 2(300 - 85 \cdot 3) - 85 = 2 \cdot 300 - 7 \cdot 85$. Поэтому $5 = 2 \cdot 300 + (-7) \cdot 85$.

Ответ: НОД$(300, 85) = 5 = 2 \cdot 300 + (-7) \cdot 85$.

Определение 3.2.5. *Общим кратным целых чисел* $a_1, a_2, \ldots, a_k$, $k \geq 2$, отличных от нуля называется целое число, которое делится на каждое из этих чисел (ОК).

Очевидно, что для любой совокупности целых чисел $a_1, a_2, \ldots a_k$, $a_i \neq 0$, $i = \overline{1, k}$, существует бесконечно много общих кратных, например числа вида $a_1 \cdot a_2 \cdot \ldots \cdot a_k \cdot n$, где $n \in \mathbb{Z}$.

Заметим, что $|a_1 \cdot a_2 \cdot \ldots \cdot a_k|$ — натуральное общее кратное совокупности целых чисел $a_1, a_2, \ldots, a_k$, $k \geq 2$, поэтому наименьшее натуральное общее кратное либо равно этому числу, либо меньше его. Если натуральное ОК меньше $|a_1 \cdot a_2 \cdot \ldots \cdot a_k|$, то оно содержится в промежутке от 1 до $|a_1 \cdot a_2 \cdot \ldots \cdot a_k|$, где находится конечное число натуральных чисел, а значит, и конечное число натуральных общих кратных совокупности $a_1, a_2, \ldots, a_k$, среди которых найдется наименьшее.

Определение 3.2.6. Наименьшее натуральное ОК целых чисел, отличных от нуля, называется *наименьшим общим кратным этих чисел* и обозначается НОК$(a_1, a_2, \ldots, a_k)$ или $[a_1, a_2, \ldots, a_k]$, $k \geq 2$.

Теорема 3.2.3. $\text{НОК}(a, b) = \dfrac{a \cdot b}{\text{НОД}(a, b)}$, $a, b \in \mathbb{N}$.

Пример 3.2.2. Найдите НОК$(300, 85)$.

По теореме (3.2.3) имеем:

$$\text{НОК}(300, 85) = \frac{300 \cdot 85}{\text{НОД}(300, 85)} = \frac{300 \cdot 85}{5} = 5\,100.$$

Ответ: НОК$(300, 85) = 5\,100$.

Реализация в системе GAP

Нахождение НОДа целых чисел в системе GAP реализуется при помощи следующих функций:

- GcdInt(a, b) возвращает НОД целых чисел a и b;
- Gcdex(a, b) возвращает запись, состоящую из НОДа целых чисел a и b, его линейного представления через a и b и линейного представления 0 через числа a и b;
- ShowGcd(a,b) возвращает НОД целых чисел a и b, а также рузультат каждого шага алгоритма Евклида;
- LcmInt(a, b) возвращает НОК целых чисел a и b.

Пример 3.2.3. Вычислите НОД$(300, 85)$. Выразите НОД через исходные числа.

```
gap> GcdInt(300,85);
5
gap> Gcdex(300,85);
rec( coeff1 := 2, coeff2 := -7, coeff3 := -17, coeff4 := 60, gcd := 5 )
```

Пример 3.2.4. Разработайте функцию, с помощью которой можно получить информацию о каждом этапе алгоритма Евклида.

Воспользуемся функцией ostatok(a,b) из примера 3.1.5.

```
euclid:=function(a,b,ostatok)
local r,q;
r:=a mod b;
q:=ostatok(a,b);
```

```
Print(a, "= ", b, "* ", q, "+ ", r, "\n ");
while r <> 0 do
a:=b;
b:=r;
r:=a mod b;
q:=ostatok(a,b);
Print(a, "= ", b, "* ", q, "+ ", r, "\n ");
od;
Print("NOD = ", b, "\n ");
return b;
end;
gap> Read("C:/gap4r7/bin/ostatok.g");
gap> Read("C:/gap4r7/bin/euclidPrint.g");
```

```
gap> euclidPrint(300,85,ostatok);
300 = 85 * 3 + 45
85 = 45 * 1 + 40
45 = 40 * 1 + 5
40 = 5 * 8 + 0
NOD = 5
```

Этот же пример можно решить, используя встроенную функцию ShowGcd(a,b).

```
gap> ShowGcd(300,85);
300=3*85 + 45
85=1*45 + 40
45=1*40 + 5
40=8*5 + 0
The Gcd is 5
= 1*45 -1*40
= -1*85 + 2*45
= 2*300 -7*85
```

Пример 3.2.5. Найдите НОК(300, 85).

```
gap> LcmInt(300,85);
5100
```

3.3 Простые числа. Разложение натуральных чисел на простые множители. Числовые функции

Определение 3.3.1. Натуральное число называется *простым*, если оно имеет только два натуральных делителя (1 и само число).

Например, числа 2, 3, 5, 7, 11, 13, 17 являются простыми числами, т.к. каждое из этих чисел имеет только два натуральных делителя.

Определение 3.3.2. Натуральное число называется *составным*, если оно имеет более двух натуральных делителей (хотя бы один натуральный делитель, отличный от 1 и самого числа).

Замечание 3.3.1. 1. Единица не является ни простым, ни составным числом, т.к. имеет только один натуральный делитель.

2. Единственным четным простым числом является число 2.

Теорема 3.3.1. (Критерий составного числа). Натуральное число $a > 1$ является составным тогда и только тогда, когда оно делится хотя бы на одно простое число, не превосходящее $\sqrt{a}$.

Теорема 3.3.2. (Критерий простого числа). Натуральное число $a > 1$ является простым тогда и только тогда, когда оно не делится ни на одно простое число p, не превосходящее $\sqrt{a}$.

Пример 3.3.1. Выясните, простым или составным является число 101.

Воспользуемся критерием простого числа. Очевидно, что $\sqrt{101} \approx 10,05$. Рассмотрим $p < 10$, т.е. $2, 3, 5, 7$. Число 101 не делится ни на одно из этих

чисел, значит, 101 — простое число.

Ответ: является.

Теорема 3.3.3. (Теорема Евклида). Множество простых чисел бесконечно.

Пример 3.3.2. Найдите возможные значения простого числа p, если известно, что $4p^2+1$ и $6p^2+1$ — простые числа.

Все натуральные числа можно представить в виде $5n$, $5n \pm 1$, $5n \pm 2$. Числа вида $5n$ являются простыми только при $n = 1$. В этом случае $p = 5$ и $4p^2+1 = 101$, $6p^2+1 = 151$, т.е. мы нашли одно значение p, удовлетворяющее условию.

Покажем, что других значений p нет. Если $p = 5n \pm 1$, то $4p^2+1 = 4(20n^2 \pm 8n + 1)$ — число составное; если $p = 5n \pm 2$, то $6p^2+1 = 5(30n^2 \pm 24n + 1)$ — число составное.

Теорема 3.3.4. (Основная теорема арифметики). Любое натуральное число $a > 1$ можно разложить на простые множители, и это разложение единственно с точностью до порядка следования множителей.

В разложении натурального числа a на простые множители могут встречаться равные множители. Пусть множитель p_1 встречается α_1 раз, p_2 — α_2 раз, ..., p_k — α_k раз. Тогда число a примет вид $a = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_n}$, $\alpha_i \in \mathbb{N}$, $i = \overline{1,k}$.

Такое разложение называется *каноническим разложением числа* a.

Пример 3.3.3. Найдите каноническое разложение числа 3 445.

Очевидно, что число 3 445 делится на 5. Тогда $3445 = 5 \cdot 689$. Т.к.

$\sqrt{689} < 30$, то нужно проверить все простые числа не более 30. Числа 2, 3, 5, 7, 11 не делят 689, а 13 делит $689 = 13 \cdot 53$. Число 53 является простым. Т.о., каноническое разложение числа 3445 имеет вид $3445 = 5 \cdot 13 \cdot 53$.

Ответ: $3445 = 5 \cdot 13 \cdot 53$.

Таблицу простых чисел, не превышающих заданного натурального числа n, можно составить следующим образом. Выпишем все натуральные числа от 2 до n:

$$2,\ 3,\ 4,\ 5,\ 6, \ldots, n.$$

Далее вычеркнем в последовательности все числа, кратные 2. Первое невычеркнутое число 3 является простым. Это число оставляем и затем вычеркиваем все числа, кратные 3. Первым невычеркнутым числом после этого будет 5, которое является простым. Его оставляем и далее вычеркиваем все числа, кратные 5, и т.д. Вычеркнув, таким образом, все числа, кратные простым числам, не превышающим $\sqrt{n}$, получим все простые числа на отрезке от 1 до n.

Данный метод выделения простых чисел называется *решетом Эратосфена* по имени древнегреческого математика, впервые использовавшего его.

Пример 3.3.4. Найдите все простые числа между 100 и 110.

Т.к. $\sqrt{109} \approx 10$, то наименьший простой делитель указанных чисел

≤ 7. Выпишем указанные числа и подчеркнем кратные 2, 3, 5 и 7: 101, 102, 103, 104, 105, 106, 107, 108, 109. Т.к. $101 = 7 \cdot 14 + 3$, то наименьшее кратное семи число — четвертое от 101, т.е. 105; оно уже подчеркнуто, а следующее кратное семи число больше 109 (седьмое от 105). Следовательно, среди указанных чисел кратных 7 нет.

Ответ: 101, 103, 107 и 109.

В теории чисел рассматриваются разнообразные функции $f(n)$, значения которых при натуральных значениях n связаны с арифметической природой n. Множество рассматриваемых функций удобнее не ограничивать заранее какими-либо требованиями, кроме единственного требования: каждая функция должна быть определена для всех натуральных значений аргумента.

Определение 3.3.3. Функция $f(x)$ называется *числовой*, если она определена при всех натуральных значениях аргумента x.

Рассмотрим сначала числовые функции $\tau(n)$ и $\sigma(n)$, зависящие от делителей аргумента. Функция $\tau(n)$ определяется как *число положительных делителей натурального числа* n, а функция $\sigma(n)$ определяется как *сумма положительных делителей натурального числа* n, т.е.

$$\tau(n) = \sum_{d|n} 1, \sigma(n) = \sum_{d|n} d.$$

Пример 3.3.5. Если p простое, то $\tau(p) = 2$. Например, $\tau(1) = 1$, $\tau(18) = 6$, т.к. у числа 18 шесть положительных делителей: 1, 2, 3, 6, 9

и 18. $\sigma(18) = 1 + 2 + 3 + 6 + 9 + 18 = 39; \sigma(p) = 1 + p$.

Теорема 3.3.5. Если $p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_s^{\alpha_s}$ — каноническое разложение натурального числа n, то

$$\tau(n) = (\alpha_1 + 1)(\alpha_2 + 1)\ldots(\alpha_s + 1).$$

Пример 3.3.6. $\tau(1\ 000\ 000) = \tau(2^6 \cdot 5^6) = 7 \cdot 7 = 49$, $\tau(48\ 510) = \tau(2 \cdot 3^2 \cdot 5 \cdot 7^2 \cdot 11) = 2 \cdot 3 \cdot 2 \cdot 3 \cdot 2 = 72$.

Теорема 3.3.6. Если $p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_s^{\alpha_s}$ — каноническое разложение натурального числа n, то

$$\sigma(n) = \frac{p_1^{\alpha_1+1} - 1}{p_1 - 1} \cdot \frac{p_2^{\alpha_2+1} - 1}{p_2 - 1} \cdot \ldots \cdot \frac{p_s^{\alpha_s+1} - 1}{p_s - 1}.$$

Пример 3.3.7. $\sigma(19\ 800) = \sigma(2^3 \cdot 3^2 \cdot 5^2 \cdot 11) = \frac{2^4-1}{2-1} \cdot \frac{3^3-1}{3-1} \cdot \frac{5^3-1}{5-1} \cdot \frac{11^2-1}{11-1} = 72\ 540$.

Пример 3.3.8. Найдите натуральное число x, если известно, что 12 делит x и $\tau(x) = 14$.

Натуральное число x записывается в виде:

$$x = 2^{\alpha} 3^{\beta} p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k},\ \alpha \geq 2,\ \beta \geq 1,$$

$$3 < p_1 < p_2 < \ldots < p_k,\ k \geq 0.$$

По условию

$$\tau(x) = (\alpha + 1)(\beta + 1)(\alpha_1 + 1)\ldots(\alpha_k + 1) = 14 = 2 \cdot 7,$$

где $\alpha+1 \geq 3$, $\beta+1 \geq 2$. Это возможно лишь в случае, когда $k=0$, $\alpha+1=7$, $\beta+1=2$ и $x=2^6 \cdot 3=192$.

Ответ: 192.

Определение 3.3.4. *Функция Эйлера* $\varphi(n)$ определена на множестве $\mathbb{N}$ и представляет собой число натуральных чисел, не превосходящих n и взаимно простых с ним.

Например, $\varphi(1)=\varphi(2)=1$, $\varphi(3)=\varphi(4)=2$ и т.д.

Теорема 3.3.7. $\varphi(p^n)=p^n(1-\frac{1}{p})=p^{n-1}(p-1)$

Следствие 3.3.1. $\varphi(p)=p-1$.

Следствие 3.3.2. Если $n=p_1^{\alpha_1} \cdot \ldots \cdot p_t^{\alpha_t}$, то

$$\varphi(n)=p_1^{\alpha_1}(1-\frac{1}{p_1}) \cdot \ldots \cdot p_t^{\alpha_t}(1-\frac{1}{p_t})=$$

$$=n(1-\frac{1}{p_1}) \cdot \ldots \cdot (1-\frac{1}{p_t})=p_1^{\alpha_1-1}(p_1-1) \cdot \ldots \cdot p_t^{\alpha_t-1}(p_t-1).$$

Пример 3.3.9. $\varphi(360)=\varphi(2^3 3^2 5)=4 \cdot 3 \cdot 2 \cdot 4=96$.

Пример 3.3.10. Найдите все простые делители числа x из уравнения $3\varphi(x)=x$.

По условию 3 делит x, значит $3-1=2$ делит $\varphi(x)$, а из равенства $3\varphi(x)=x$ следует, что x делится на 6. Будем считать, что

$$x=2^{\alpha}3^{\beta}p_1^{\alpha_1}\ldots p_k^{\alpha_k},\ 3<p_1<\ldots<p_k,\ k \geq 0.$$

Предположим, что $k > 0$. По условию

$$3 \cdot 2^{\alpha-1} \cdot 3^{\beta-1} \cdot 2 \cdot p_1^{\alpha_1-1}(p_1-1)\dots p_k^{\alpha_k-1}(p_k-1) = 2^{\alpha}3^{\beta}p_1^{\alpha_1}\dots p_k^{\alpha_k},$$

поэтому $(p_1-1)\dots(p_k-1) = p_1\dots p_k$. Так как $p_1 - 1 < \dots < p_k - 1 < p_k$, то p_k делит $(p_1-1)\dots(p_k-1)$, что невозможно. Поэтому допущение $k > 0$ неверно. Значит, $k = 0$ и $x = 2^{\alpha}3^{\beta}$.

Ответ: 2; 3.

Пример 3.3.11. Решите уравнение $\varphi(3^x5^y) = 40$.

Т.к. $\varphi(3^x5^y) = 3^{x-1}(3-1)5^{y-1}(5-1) = 40 = 2^3 5$, то $3^{x-1}5^{y-1} = 5$. Поэтому $x = 1$, а $y = 2$.

Ответ: $(1;2)$.

Реализация в системе GAP

Исследование натуральных чисел на простоту и работа с основными числовыми функциями в системе компьютерной алгебры GAP реализуется при помощи следующих функций:

- Primes возвращает список всех простых чисел, не превосходящих 1 000;
- IsPrimeInt(a) возвращает результат true, если натуральное число a простое, и результат false, если число a составное;
- FactorsInt(a) возвращает список простых делителей заданного целого числа a.

• PrimePowersInt(a) возвращает каноническое разложение целого числа $a = p_1^{\alpha_1} \cdot \ldots \cdot p_t^{\alpha_t}$ в виде $[p_1, \alpha_1, ..., p_t, \alpha_t]$;

• PrintFactorsInt(a) возвращает каноническое разложение целого числа a;

• Sigma(a) возвращает значение функции $\sigma(a)$;

• Tau(a) возвращает значение функции $\tau(a)$;

• Phi(a) возвращает значение функции Эйлера $\varphi(a)$.

Пример 3.3.12. Найдите все простые числа, не превосходящие 1000.

```
gap> Primes;
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
  71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139,
  149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211,
  223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281,
  283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367,
  373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443,
  449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523,
  541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607,
613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683,
  691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773,
  787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863,
  877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967,
  971, 977, 983, 991, 997 ]
```

```
gap> Primes[1];
2
gap> Primes[8];
19
```

Пример 3.3.13. Выясните, простыми или составными является число 101.

```
gap> IsPrimeInt(101);
true
```

Пример 3.3.14. Разложите числа 3 445 и 12! на простые множители.

```
gap> FactorsInt(3445);
[5, 13, 53]
gap> PrimePowersInt(3445);
[5, 1, 13, 1, 53, 1]
gap> PrintFactorsInt(3445);
5*13*53
gap> FactorsInt(Factorial(12));
[ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 5, 5, 7, 11 ]
gap> PrimePowersInt(Factorial(12));
[ 2, 10, 3, 5, 5, 2, 7, 1, 11, 1 ]
gap> PrintFactorsInt(Factorial(12));
2^10 * 3^5 * 5^2 * 7 * 11
```

Пример 3.3.15. Найдите $\tau(1\ 000\ 000)$ и $\sigma(19\ 800)$.

```
gap> DivisorsInt(19800);
[ 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 15, 18, 20, 22, 24, 25, 30, 33, 36,
  40, 44, 45, 50, 55, 60, 66, 72, 75, 88, 90, 99, 100, 110, 120, 132,
  150, 165, 180, 198, 200, 220, 225, 264, 275, 300, 330, 360, 396,
  440, 450, 495, 550, 600, 660, 792, 825, 900, 990, 1100, 1320, 1650,
  1800, 1980, 2200, 2475, 3300, 3960, 4950, 6600, 9900, 19800 ]
gap> Sigma(19800);
72540
gap> Sum(DivisorsInt(19800))=Sigma(19800);
true
gap> DivisorsInt(1000000);
[ 1, 2, 4, 5, 8, 10, 16, 20, 25, 32, 40, 50, 64, 80, 100, 125, 160, 200,
  250, 320, 400, 500, 625, 800, 1000, 1250, 1600, 2000, 2500, 3125,
  4000, 5000, 6250, 8000, 10000, 12500, 15625, 20000, 25000,
  31250, 40000, 50000, 62500, 100000, 125000, 200000, 250000,
  500000, 1000000 ]
gap> Tau(1000000);
49
gap> Length(DivisorsInt(1000000))=Tau(1000000);
true
```

Пример 3.3.16. Найдите натуральное число x, если известно, что 12 делит x и $\tau(x) = 14$.

```
gap> Filtered([1..50000], x-> Tau(x) = 14 and x mod 12 = 0);
[ 192 ]
```

Пример 3.3.17. Найдите $\varphi(360)$.

```
gap> Phi(360);
96
```

Пример 3.3.18. Разработайте функцию, которая для любых целых a и b выдавала все простые делители числа x из уравнения $a\varphi(x) = bx$ на интервале [1;1000]. Найдите все простые делители числа x из уравнения $3\varphi(x) = x$.

```
uravn:=function(a,b)
local i,k,s;
for i in [1..1000] do
if a*Phi(i)=b*i then
k:=i;
break;  fi;
od;
s:=Set(Factors(k));
return s;
end;
```

```
gap> Read("C:/gap4r7/bin/uravn.g");
gap> uravn(3,1);
[ 2, 3 ]
```

3.4 Отношение сравнения в кольце $\mathbb{Z}$

Теорема 3.4.1. Пусть m — натуральное число, $m > 1$. Для любых целых чисел a и b следующие утверждения равносильны:

1) a и b имееют одинаковые остатки от деления на m;

2) $a - b$ делится на m, т.е. $a - b = mq$ для подходящего целого q;

3) $a = b + mq$ для некоторого целого q.

Определение 3.4.1. Целые числа a и b называются *сравнимыми по модулю* m, если они удовлетворяют одному из условий теоремы 3.4.1, и пишут $a \equiv b \pmod{m}$. Данное соотношение между целыми числами называют сравнением по модулю m.

Определение 3.4.2. *Сравнением первой степени с одним неизвестным* называется сравнение $ax \equiv b \pmod{m}$, где $a, b \in \mathbb{Z}$, $m \in \mathbb{N}$, $a \not\equiv 0 \pmod{m}$.

Если в это сравнение вместо x будем подставлять различные целые числа, то будем получать верные или неверные числовые сравнения. Те значения x, которые дают верные числовые сравнения, называют *решениями сравнения*.

Легко проверить, что если x_0 — решение сравнения, то все целые числа из класса $\overline{x_0} = \{x_0 + tm \mid t \in \mathbb{Z}\}$ также будут решениями. Такие решения считаются одинаковыми. Поэтому решением сравнения принято считать не отдельное число, а целый класс вычетов по модулю m, удовлетворяющих сравнению.

Определение 3.4.3. *Числом решений сравнения* называют число решений сравнения в какой-либо полной системе вычетов по модулю m.

Определение 3.4.4. Сравнения называются *равносильными*, если они имеют одинаковые решения.

Теорема 3.4.2. 1.Если НОД$(a, m) = 1$, то сравнение имеет единственное решение.

2. Если НОД$(a, m) = d > 1$ и d не делит b, то сравнение не имеет решений.

3. Если НОД$(a, m) = d > 1$ и d делит b, то сравнение имеет d решений по модулю m: $\overline{x_0}$, $\overline{x_0 + 2m_1}$, $\ldots$, $\overline{x_0 + (d-1)m_1}$, где $m_1 = \dfrac{m}{d}$, x_0 — наименьший неотрицательный вычет из решения сравнения

$$a_1 x = b_1 \pmod{m_1}, a_1 = \frac{a}{d}, b_1 = \frac{b}{d}.$$

Пример 3.4.1. Решите сравнение $5x \equiv 6 \pmod 7$.

1. Метод проб. Т.к. НОД$(5, 7) = 1$, то по теореме 3.4.2 (1) сравнение имеет единственное решение. Подставляя наименьшие по абсолютной величине вычеты 0, ± 1, ± 2, ± 3 по модулю 7 в сравнение, получаем, что

$\overline{4} \in \mathbb{Z}_7$ — искомое решение сравнения.

Ответ: $\overline{4}$.

2. Метод преобразования коэффициентов: используя свойства сравнений, коэффициенты сравнения преобразуют так, чтобы коэффициент при x стал равен 1.

В сравнении к левой части прибавим $-7x$:

$$-2x \equiv 6 \pmod 7.$$

Т.к. НОД$(-2, 7) = 1$, то разделим обе части последнего сравнения на (-2):

$$x \equiv -3 \equiv 4 \pmod 7.$$

3. Метод Эйлера: решение находится по формуле

$$x \equiv ba^{\varphi(m)-1} \pmod m.$$

Решим исходное сравнение методом Эйлера. Т.к. $\varphi(7) = 6$, то

$$x \equiv 6 \cdot 5^{6-1} \equiv 6 \cdot 5^3 \cdot 5^2 \equiv (-1)(-1) \cdot 4 \equiv 4 \pmod 7.$$

Пример 3.4.2. Решите сравнение $45x \equiv 31 \pmod{100}$.

Т.к. НОД$(45, 100) = 5$ и 31 не делится на 5, то по теореме 3.4.2 (2) сравнение решений не имеет.

Ответ: решений нет.

Пример 3.4.3. Решите сравнение $51x \equiv 141 \pmod{234}$.

Здесь НОД$(51, 234) = 3$ и 141 делится на 3. Следовательно, сравнение имеет 3 решения по теореме 3.4.2 (3).

После деления обеих частей сравнения и модуля на 3 получим сравнение $17x \equiv 47 \pmod{78}$. Полученное сравнение имеет единственное решение, т.к. НОД$(17, 78) = 1$. Его решением является $x \equiv 67 \pmod{78}$ (проверьте). Тогда $\overline{67}, \overline{145}, \overline{223}$ — решения данного сравнения.

Ответ: $\overline{67}, \overline{145}, \overline{223}$.

Рассмотрим систему сравнений специального вида

$$\begin{cases} x \equiv a_1 \pmod{m_1}, \\ \dots\dots\dots\dots\dots \\ x \equiv a_k \pmod{m_k}. \end{cases} \tag{3.4.1}$$

Теорема 3.4.3. (Китайская теорема об остатках). Если $m_1, \dots, m_k$ попарно взаимно просты, то система сравнений (3.4.1) разрешима. Определим целые числа M, M_i, b_i условиями $M = m_1 m_2 \cdot \dots \cdot m_k$, $M_i = \frac{M}{m_i}$, $M_i b_i \equiv a_i \pmod{m_i}$, $i = 1, 2, \dots, k$,

$$x_0 = M_1 b_1 + M_2 b_2 + \dots + M_k b_k.$$

Тогда множество целых чисел, удовлетворяющих системе сравнений, составляет класс вычетов $x \equiv x_0 \pmod{M}$.

Замечание 3.4.1. Поскольку в условиях теоремы 3.4.3 НОД$(M_i, m_i) = 1$, существование чисел b_i следует из теоремы 3.4.2. Заметим также, что числа b_i определяются не единственным способом. При использовании теоремы 3.4.3 для решения систем сравнений следует выбирать те из них, которые дают по возможности меньшие значения x_0.

Пример 3.4.4. Решите систему сравнений

$$\begin{cases} x \equiv 4 \pmod 5, \\ x \equiv 1 \pmod{12}, \\ x \equiv 7 \pmod{14}. \end{cases}$$

Из первого сравнения имеем: $x = 5t + 4$.

Подставляем во второе сравнение: $5t+4 \equiv 1 \pmod{12}$, $5t \equiv 9 \pmod{12}$, откуда $t \equiv 9 \pmod{12}$, $t = 12t_1 + 9$. Подставляя найденное значение t в равенство $x = 5t + 4$, находим: $x = 5(12t_1 + 9) + 4 = 60t_1 + 49$. Найденное значение x подставляем в третье сравнение: $60t_1 + 49 \equiv 7 \pmod{14}$, $60t_1 \equiv {}^{-}42 \pmod{14}$, $4t_1 \equiv 0 \pmod{14}$. Делим обе части сравнения и модуль на 2: $2t_1 \equiv 0 \pmod 7$, $t_1 \equiv 0 \pmod 7$, откуда $t_1 = 7t_2$. Подставляя найденные значения t_1 в равенство $x = 60t_1 + 49$, находим: $x = 60 \cdot 7t_2 + 49 = 420t_2 + 49$.

Выполним проверку: $49 - 4$ делится на 5; $49 - 1$ делится на 12; $49 - 7$ делится на 14.

Ответ: $x \equiv 49 \pmod{2^2 \cdot 3 \cdot 5 \cdot 7}$.

Пример 3.4.5. Решите систему сравнений

$$\begin{cases} x \equiv 20 \pmod{21}, \\ x \equiv 3 \pmod{5}, \\ x \equiv 5 \pmod{8}. \end{cases}$$

Здесь $M = 21 \cdot 5 \cdot 8 = 840$, $M_1 = M/m_1 = 840/21 = 40$, $M_2 = 168$, $M_3 = 105$. Решаем сравнения:

$40x \equiv 20 \pmod{21}$, $\quad x = 11 = b_1$,
$168x \equiv 3 \pmod{5}$, $\quad x = 1 = b_2$,
$105x \equiv 5 \pmod{8}$, $\quad x = 5 = b_3$.

Вычисляем значение $x_0 = M_1 b_1 + M_2 b_2 + M_3 b_3$:

$$x_0 = 40 \cdot 11 + 168 \cdot 1 + 105 \cdot 5 = 1133.$$

Тогда $x \equiv 1133 \equiv 293 \pmod{840}$.

Ответ: $x \equiv 293 \pmod{840}$.

Реализация в системе GAP

Решить вопрос о сравнимости двух целых чисел, используя компьютерную систему GAP, можно следующим образом:

Пример 3.4.6. Проверьте, сравнимы ли числа 35 323 252 и 345 343 по модулю 17?

```
gap> a:=35323252;
35323252
gap> b:=345343;
345343
gap> a mod 17=b mod 17;
false
gap> IsInt((a-b)/17);
false
```

Пример 3.4.7. Основываясь на теореме 3.4.2, составьте в системе GAP программу, решающую произвольное сравнение первой степени $ax \equiv b \pmod{m}$.

```
CongrSol:=function(a,b,m)
local sol, d, a1, b1, m1, x0, k;
sol:=[];
d:=Gcd(a,m);
if d>1 and not IsInt(b/d) then
return [];
else
if d=1 then
```

```
Add(sol, (b*(a^(Phi(m)-1))) mod m);
return sol;
else
a1:=a/d; b1:= b/d; m1:=m/d;
x0:=CongrSol(a1, b1, m1);
if x0 <> [] then
for k in [0..d-1] do
Add(sol, k*m1+x0);
od;
fi;
return sol;
fi;
fi;
return sol;
end;
```

Пример 3.4.8. Решите сравнения в системе GAP: $5x \equiv 6 \pmod 7$, $45x \equiv 31 \pmod{100}$, $51x \equiv 141 \pmod{234}$.

```
gap> Read("C:/gap4r7/bin/CongrSol.g");
gap> CongrSol(45,31,100);
[ ]
```

```
gap> CongrSol(51,141,234);
[ [ 67 ], [ 145 ], [ 223 ] ]
gap> CongrSol(5,6,7);
[ 4 ]
```

Т.к. решением сравнения первой степени принято считать класс вычетов по модулю m, то всякое сравнение можно свести к решению уравнения первой степени с коэффициентами из кольца классов вычетов $\mathbb{Z}_m$. Будем в дальнейшем использовать следующие функции:

- ZmodnZ(m) возвращает кольцо классов вычетов $\mathbb{Z}_m$;
- ZmodnZObj(r,m) возвращает класс вычетов $\overline{r}$ из кольца $\mathbb{Z}_m$;
- ChineseRem([m1,m2,...,mk], [a1,a2,..., ak]) возвращает решение системы сравнений вида (3.4.1).

```
gap> r:= ZmodnZ(15);
(Integers mod 15)
gap> a:= ZmodnZObj(9,15);
ZmodnZObj( 9, 15 )
gap> b:= ZmodnZObj(13,15);
ZmodnZObj( 13, 15 )
gap> c:=a+b;
ZmodnZObj( 7, 15 )
```

Пример 3.4.9. Напишите в системе GAP программу, решающую уравнение $ax = b$, где $a, b \in \mathbb{Z}_m$.

```
CongrSol1:=function(a,b,m)
local x,i,otv;
otv:=[];
for i in [0..m-1] do
x:=ZmodnZObj(i,m);
if ZmodnZObj(a,m)*x=ZmodnZObj(b,m) then
Add(otv,x);
fi; od;
return otv; end;
gap> Read("C:/gap4r7/bin/CongrSol1.g");
gap> CongrSol1(45,31,100);
[ ]
gap> CongrSol1(51,141,234);
[ ZmodnZObj( 67, 234 ), ZmodnZObj( 145, 234 ), ZmodnZObj( 223,
  234 ) ]
gap> CongrSol1(5,6,7);
[ Z(7)^4 ]
gap> List(last,Int);
[ 4 ]
```

Пример 3.4.10. Решите системы сравнений $\begin{cases} x \equiv 4 \pmod 5, \\ x \equiv 1 \pmod{12}, \\ x \equiv 7 \pmod{14} \end{cases}$ и

$$\begin{cases} x \equiv 20 \pmod{21}, \\ x \equiv 3 \pmod 5, \\ x \equiv 5 \pmod 8. \end{cases}$$

```
gap> ChineseRem([5,12,14],[4,1,7]);
49
gap> ChineseRem([21,5,8],[20,3,5]);
293
```

Данную задачу можно решить, не используя встроенные функции.

```
CongrSysSol:=function(a1,b1,m1,a2,b2,m2,a3,b3,m3)
local i, j,otv,c;
otv:=[];
c:=LcmInt(m1,LcmInt(m2,m3));
for i in [1..c] do
if ZmodnZObj(a1,m1)*ZmodnZObj(i,m1)=ZmodnZObj(b1,m1) and
 ZmodnZObj(a2,m2)*ZmodnZObj(i,m2)=ZmodnZObj(b2,m2) and
 ZmodnZObj(a3,m3)*ZmodnZObj(i,m3)=ZmodnZObj(b3,m3) then
Add(otv,ZmodnZObj(i,c));
fi; od;
```

```
return otv;
end;
gap> Read("C:/gap4r7/bin/CongrSysSol.g");
CongrSysSol(7, 3, 11, 3, 1, 7, 3, 2, 5);
[ ZmodnZObj( 299, 385 ) ]
gap> CongrSysSol(1, 13, 16, 1, 3, 10, 1, 9, 14);
[ ZmodnZObj( 93, 560 ) ]
```

3.5 Порядок числа по данному модулю

Пусть НОД$(a, m) = 1$, $a \in \mathbb{Z}, m \in \mathbb{N}$.

Определение 3.5.1. *Порядком (показателем) числа a по модулю m* называется наименьшее натуральное число k такое, что $a^k \equiv 1 \pmod{m}$ и обозначается через $\theta(a \mod m)$.

Пример 3.5.1. Найдите $\theta(2 \mod 15)$.

$2^1, 2^2, 2^3 \not\equiv 1 \pmod{15}, 2^4 \equiv 1 \pmod{15}$, значит, $\theta(2 \mod 15) = 4$.

Ответ: 4.

Определение 3.5.2. Если $\theta(a \mod m) = \varphi(m)$, то a называется *первообразным корнем по модулю m*.

Пример 3.5.2. Какой порядок имеет число 5 по модулю 12?

Должны быть выполнены следующие требования:

а) искомый порядок надо искать среди делителей числа $\varphi(m)$, где m

— модуль;

б) искомый порядок должен быть наименьшим из положительных показателей, удовлетворяющих сравнению $a^z \equiv 1 \pmod{m}$, где a – испытуемое число.

В данном случае имеем:

$$\text{НОД}(5, 12) = 1;\ \varphi(12) = 12 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{3}\right) = 4.$$

Делителями 4 являются числа 1, 2, 4. Тогда

$$5^1 \equiv 5 \pmod{12},\ 5^2 \equiv 1 \pmod{12}.$$

Следовательно, число 5 имеет порядок 2 по модулю 12.

Ответ: 2.

Пример 3.5.3. Какой порядок имеет число 4 по модулю 12?

Числа 4 и 12 не являются взаимно простыми, а следовательно, сама постановка вопроса является ошибочной.

Пример 3.5.4. Найти наименьший первообразный корень по модулю 7.

Для нахождения наименьшего первообразного корня по простому модулю p необходимо и достаточно:

а) найти все различные простые делители числа $p - 1$ (обозначим их $p_1, p_2, \ldots, p_k$);

б) последовательно проверить числа, взаимно простые с модулем, начиная с числа 1; первое из чисел, которое не удовлетворяет ни одному из сравнений

$$q^{\frac{p-1}{p_1}} \equiv 1 \pmod{p}, \ldots, q^{\frac{p-1}{p_k}} \equiv 1 \pmod{p}$$

будет искомым первообразным корнем.

Имеем $7-1=6=2\cdot 3$. Т.к. $1^2 \equiv 1 \pmod 7$, то число 1 не является первообразным корнем по модулю 7.

Т.к. $2^2 \equiv 4 \pmod 7$, $2^3 \equiv 1 \pmod 7$, то число 2 не является первообразным корнем по модулю 7.

Т.к. $3^2 \equiv 2 \pmod 7$, $3^3 \equiv -1 \pmod 7$, то число 3 — наименьший первообразный корень по модулю 7.

Ответ: 3.

Пусть a — первообразный корень по простому модулю p.

Определение 3.5.3. Если $a^k \equiv b \pmod p$, где k — целое неотрицательное число, то число k называется *индексом числа b по модулю p и первообразному корню (основанию) a* и обозначается $k = \operatorname{ind}_a b$ или $k = \operatorname{ind} b$. Таким образом, $a^{\operatorname{ind}_a b} \equiv b \pmod p$.

Определение 3.5.4. Если сравнение $x^n \equiv c \pmod p$ имеет решения, то c называется *вычетом степени n по простому модулю p*, в противном случае — *невычетом степени n*. Вычет (невычет) называется *квадратичным* при $n=2$, *кубическим* при $n=3$, *биквадратным* при

$n = 4$.

Определение 3.5.5. Пусть p — простое число, $p > 2$ и p не делит a. *Символ Лежандра* $\left(\frac{a}{p}\right)$ задается следующим образом:

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{если } a \text{ – квадратичный вычет по модулю } p, \\ -1, & \text{если } a \text{ – квадратичный невычет по модулю } p. \end{cases}$$

Другими словами, $\left(\frac{a}{p}\right)$ равно 1, если сравнение $x^2 \equiv a \pmod{p}$ имеет два решения, и $\left(\frac{a}{p}\right)$ равно -1, если это сравнение не имеет .

Пример 3.5.5. $\left(\frac{3}{11}\right) = 1$, т.к. сравнение $x^2 \equiv 3 \pmod{11}$ имеет два решения: $x = \pm 5 \pmod{11}$; $\left(\frac{2}{5}\right) = -1$, т.к. сравнение $x^2 \equiv 2 \pmod{5}$ не имеет решений.

Реализация в системе GAP

Найти порядок элемента по модулю, указать все первообразные корни по модулю, найти индекс и символ Лежандра в системе GAP можно, используя следующие функции:

• OrderMod(a,m) возвращает порядок (показатель) числа a по модулю m. Если НОД(a, m) <> 1, то OrderMod(a,m) возвращает 0;

• PrimitiveRootMod(m[,start]) возвращает наименьший первообразный корень по модулю m. Аргумент start является целым числом, больше которого должен быть первообразный корень по модулю m;

• IsPrimitiveRootMod(r,m) возвращает true, если r является первообразным корнем по модулю m, и false в противном случае;

• LogMod(b,a,p) возвращает индекс числа b по модулю p и первообразному корню (основанию) a;

• Legendre(n, m) возвращает Символ Лежандра целого числа n по модулю m.

Пример 3.5.6. Какие порядки имеют числа 5 и 4 по модулю 12?

```
gap> OrderMod(5,12);
2
gap> OrderMod(4,12);
0
```

Пример 3.5.7. Найдите наименьший первообразный корень по модулю 7.

```
gap> PrimitiveRootMod(7);
3
```

Пример 3.5.8. Найдите все попарно несравнимые первообразные

корни по модулю 17.

```
gap> PrimitiveRootMod(17,1);
3
gap> PrimitiveRootMod(17,3);
5
gap> PrimitiveRootMod(17,5);
6
gap> PrimitiveRootMod(17,6);
7
gap> PrimitiveRootMod(17,7);
10
gap> PrimitiveRootMod(17,10);
11
gap> PrimitiveRootMod(17,11);
12
gap> PrimitiveRootMod(17,12);
14
gap> PrimitiveRootMod(17,14);
fail
```

Пример 3.5.9. Найдите индекс числа 2 по модулю 7 и первообразному корню (основанию) 5.

```
gap> IsPrimitiveRootMod(5,7);
true
gap> l:= LogMod( 2, 5, 7 );
4
5^4 mod 7 = 2;
true
```

Пример 3.5.10. Найдите $\left(\frac{3}{11}\right)$ и $\left(\frac{2}{5}\right)$.

```
gap> Legendre(3,11);
1
gap> Legendre(2,5);
-1
```

РАЗДЕЛ 4
ЭЛЕМЕНТЫ ЛИНЕЙНОЙ АЛГЕБРЫ В СИСТЕМЕ GAP

4.1 Матрицы и операции над ними

Пусть $\mathbb{P}$ — числовое поле, а m и n — натуральные числа.

Определение 4.1.1. Прямоугольная таблица

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ & & \dots & & & \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ & & \dots & & & \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{bmatrix}, i = \overline{1,m}, j = \overline{1,n},$$

составленная из элементов поля $\mathbb{P}$, называется *матрицей размера m на n* или $m \times n$-матрицей над полем $\mathbb{P}$, а числа a_{ij} называются элементами этой матрицы. Если $m = n$, то A называется *квадратной матрицей порядка n*. Матрицу A можно записать сокращенно: $A = [a_{ij}], i = \overline{1,m}, j = \overline{1,n}$.

Матрицы $A = [a_{ij}]$ и $B = [b_{ij}]$ называются *равными*, если они имеют одинаковую размерность и соответствующие элементы этих матриц совпадают, т.е. $a_{ij} = b_{ij}$ для всех i и j.

Определение 4.1.2. Матрица называется *нулевой*, если все ее элементы равны нулевому элементу поля $\mathbb{P}$. Квадратная матрица порядка

n называется *единичной*, если

$$E_n = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ & & & \ldots & & \\ 0 & 0 & 0 & \ldots & 1 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}.$$

Здесь 1 — единичный элемент поля $\mathbb{P}$.

Определение 4.1.3. Квадратную $n \times n$-матрицу, у которой на диагонали находятся элементы $a_1, a_2, \ldots, a_{n-1}, a_n$ из поля $\mathbb{P}$, а вне диагонали — нулевой элемент поля $\mathbb{P}$, называется *диагональной*:

$$\begin{bmatrix} a_1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & a_2 & 0 & \ldots & 0 & 0 \\ & & & \ldots & & \\ 0 & 0 & 0 & \ldots & a_{n-1} & 0 \\ 0 & 0 & 0 & \ldots & 0 & a_n \end{bmatrix}.$$

Если $a_1 = a_2 = \ldots = a_{n-1} = a_n = a$, то диагональную матрицу называют *скалярной* и обоначают через aE_n.

Определение 4.1.4. *Верхней треугольной матрицей* называют квад-

ратную матрицу порядка n

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n-1} & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n-1} & a_{2n} \\ & & & \dots & & \\ 0 & 0 & 0 & \dots & a_{n-1n-1} & a_{n-1n} \\ 0 & 0 & 0 & \dots & 0 & a_{nn} \end{bmatrix},$$

а *нижней треугольной матрицей* матрицу

$$\begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 & 0 \\ a_{21} & a_{22} & 0 & \dots & 0 & 0 \\ & & & \dots & & \\ a_{n-11} & a_{n-12} & a_{n-13} & \dots & a_{n-1n-1} & 0 \\ a_{n-11} & a_{n-12} & a_{n-13} & \dots & a_{n-1n-1} & a_{nn} \end{bmatrix}.$$

Определение 4.1.5. *Ступенчатой* называется матрица $A = [a_{ij}]$, обладающая следующими двумя свойствами:

1) если i-я строка нулевая, то $(i+1)$-я строка также нулевая;

2) если первые ненулевые элементы i-й и $(i+1)$-й строк располагаются в столбцах с номерами k и l соответственно, то $k < l$.

Заметим, что в определении ступенчатой матрицы не требуется, чтобы она была квадратной.

Определение 4.1.6. К *элементарным преобразованиям строк матрицы* A относятся следующие преобразования:

1) умножение какой-либо строки матрицы A на ненулевой элемент поля $\mathbb{P}$;

2) прибавление к строке матрицы A другой ее строки, умноженной на элемент поля $\mathbb{P}$.

Теорема 4.1.1. Любую ненулевую матрицу с помощью элементарных преобразований строк можно привести к ступенчатому виду.

Определение 4.1.7. *Суммой $m \times n$-матриц* A и B называется такая $m \times n$-матрица C, у которой каждый элемент $c_{ij} = a_{ij} + b_{ij}$ для всех i и j, т.е.

$$C = A + B = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ & \dots & \\ a_{m1} & \dots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & \dots & b_{1n} \\ & \dots & \\ b_{m1} & \dots & b_{mn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11} + b_{11} & \dots & a_{1n} + b_{1n} \\ & \dots & \\ a_{m1} + b_{m1} & \dots & a_{mn} + b_{mn} \end{bmatrix}.$$

Обозначим через $M_{m,n}(\mathbb{P})$ совокупность всех $m \times n$-матриц с элементами из $\mathbb{P}$.

В теореме 4.1.2. отмечены основные свойства сложения матриц.

Теорема 4.1.2 Для любых матриц $A, B, C \in M_{m,n}(\mathbb{P})$ справедливы следующие утверждения:

1) $A + B = B + A$, т.е. сложение матриц коммутативно;

2) $(A+B)+ = A+(B+C)$, т.е. сложение матриц ассоциативно;

3) $A+O=O+A=A$, где O — нулевая матрица.

Определение 4.1.8. *Произведением элемента* $u \in \mathbb{P}$ *и* $m \times n$*-матрицы* A называется такая $m \times n$-матрица D, что каждый ее элемент $d_{ij} = ua_{ij}$ для всех i и j, т.е.

$$D = u \begin{bmatrix} a_{11} & \dots & a_{1n} \\ & \dots & \\ a_{m1} & \dots & a_{mn} \end{bmatrix} = \begin{bmatrix} ua_{11} & \dots & ua_{1n} \\ & \dots & \\ ua_{m1} & \dots & ua_{mn} \end{bmatrix}.$$

В теореме 4.1.3 перечислены основные свойства умножения матрицы на число.

Теорема 4.1.3. Для любых матриц $A, B \in M_{m,n}(\mathbb{P})$ и любых $u, v \in \mathbb{P}$ справедливы следующие утверждения:

1) $1 \cdot A = A$, $0 \cdot A = O$, где O — нулевая матрица, 1 — единичный элемент поля $\mathbb{P}$;

2) $A + (-1)A = (-1)A + A = O$;

3) $(uv)A = u(vA)$;

4) $u(A+B) = uA + uB$;

5) $(u+v)A = uA + vA$.

Матрица $(-1)A$ называется *противоположной* к матрице A и обозначается $-A$.

Определение 4.1.9. *Произведением $m \times k$-матрицы A и $k \times n$-матрицы B* называется $m \times n$-матрица

$$AB = \begin{bmatrix} A_1B^1 & A_1B^2 & \dots & A_1B^n \\ A_2B^1 & A_2B^2 & \dots & A_2B^n \\ & & \dots & \\ A_mB^1 & A_mB^2 & \dots & A_mB^n \end{bmatrix},$$

где $A_iB^j = (a_{i1}a_{i2}...a_{ik})[b_{1j}b_{2j}...b_{kj}] = a_{i1}b_{1j} + a_{i2}b_{2j} + ... + a_{ik}b_{kj}$ — произведение i-ой строки матрицы A и j-го столбца матрицы B. Заметим, что умножать матрицы можно только при условии, что число столбцов первого сомножителя равно числу строк второго сомножителя.

В теореме 4.1.4 указаны основные свойства умножения матриц.

Теорема 4.1.4. 1. Если A — $m \times n$-матрица, то $AE_n = E_mA = A$.

2. Если для матриц A, B и C определено одно из произведений $(AB)C$, $A(BC)$, то определено и другое, и $(AB)C = A(BC)$. Т.е. умножение матриц ассоциативно.

3. Умножение матриц дистрибутивно относительно сложения:

— если для матриц A, B и C определено одно из выражений $(A + B)C$, $AC + BC$, то определено и другое, и $(A + B)C = AC + BC$;

— если для матриц A, B и C определено одно из выражений $A(B + C)$, $AB + AC$, то определено и другое, и $A(B + C) = AB + AC$.

4. Для любого $u \in \mathbb{P}$ и любых матриц A и B, для которых умножение

AB определено, справедливо $u(AB) = (uA)B = A(uB)$.

Определение 4.1.10. Квадратная матрица A порядка n называется *обратимой*, если существует матрица A^{-1} такая, что $AA^{-1} = A^{-1}A = E_n$. В этом случае матрица A^{-1} называется *обратной к матрице* A.

Теорема 4.1.5. Если матрица A обратима, то с помощью элементарных преобразований ее можно превратить в единичную матрицу. Если эти же преобразования в том же порядке применить к единичной матрице, то получим обратную матрицу A^{-1}.

Из теоремы 4.1.5 следует алгоритм нахождения матрицы, обратной к матрице A:

1) к матрице A дописать справа единичную матрицу E (получим матрицу $(A|E)$);

2) с помощью элементарных преобразований над матрицей $(A|E)$ прийти к матрице $(E|A^{-1})$.

Пример 4.1.1. С помощью элементарных преобразований найдите обратную матрицу для матрицы

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 2 & 3 & -3 \\ -3 & -3 & 2 \end{bmatrix}.$$

Для вычисления обратной матрицы запишем матрицу $(A|E)$, где E — единичная матрица, и преобразуем ее с помощью элементарных преобразований к матрице вида $(E|A^{-1})$.

$$(A|E) = \left[\begin{array}{ccc|ccc} 1 & 1 & -1 & 1 & 0 & 0 \\ 2 & 3 & -3 & 0 & 1 & 0 \\ -3 & -3 & 2 & 0 & 0 & 1 \end{array}\right].$$

Далее от второй строки отнимем первую строку, умноженную на 2, а к третьей строке добавим первую, умноженную на 3. Получим:

$$\left[\begin{array}{ccc|ccc} 1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & -1 & -2 & 1 & 0 \\ 0 & 0 & -1 & 3 & 0 & 1 \end{array}\right].$$

Теперь от первой строки отнимаем вторую:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 3 & -1 & 0 \\ 0 & 1 & -1 & -2 & 1 & 0 \\ 0 & 0 & -1 & 3 & 0 & 1 \end{array}\right].$$

Третью строку умножим на -1:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 3 & -1 & 0 \\ 0 & 1 & -1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -3 & 0 & -1 \end{array}\right].$$

Третью строку прибавим ко второй:

$$(E|A^{-1}) = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 3 & -1 & 0 \\ 0 & 1 & 0 & -5 & 1 & -1 \\ 0 & 0 & 1 & -3 & 0 & -1 \end{array}\right].$$

Ответ: $A^{-1} = \begin{bmatrix} 3 & -1 & 0 \\ -5 & 1 & -1 \\ -3 & 0 & -1 \end{bmatrix}$.

Определение 4.1.11. Матрица, полученная из исходной заменой строк на столбцы, называется *транспонированной*.

Другими словами, танспонированной матрицей называется матрица, у которой $i-$ столбец совпадает с i-ой строкой матрицы A при любом $i = 1, 2, 3, ..., m$. Матрица, транспонированная к матрице A, обозначается через A^T.

Теорема 4.1.6. Пусть $A = [a_{ij}]$, $B = [b_{ij}]$ — $m \times n$-матрицы, $u \in \mathbb{P}$. Тогда справедливы следующие утверждения:

1) $(A^T)^T = A$;
2) $u(A^T) = (uA)^T$;
3) $(A + B)^T = A^T + B^T$;
4) если $m = n$, то $(AB)^T = B^T A^T$.

Пример 4.1.2. Вычислите значение многочлена $A^2 + 4A - 2E_3$, если

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 2 & 3 & -3 \\ -3 & -3 & 2 \end{bmatrix}.$$

$$C = \begin{bmatrix} 1 & 1 & -1 \\ 2 & 3 & -3 \\ -3 & -3 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & -1 \\ 2 & 3 & -3 \\ -3 & -3 & 2 \end{bmatrix} + 4 \cdot \begin{bmatrix} 1 & 1 & -1 \\ 2 & 3 & -3 \\ -3 & -3 & 2 \end{bmatrix} -$$

$$-2\cdot\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}=\begin{bmatrix}6&7&-6\\17&20&-17\\-15&-18&16\end{bmatrix}+\begin{bmatrix}4&4&-4\\8&12&-12\\-12&-12&8\end{bmatrix}-$$

$$-\begin{bmatrix}2&0&0\\0&2&0\\0&0&2\end{bmatrix}=\begin{bmatrix}8&11&-10\\25&30&-29\\-27&-30&22\end{bmatrix}.$$

Ответ: $\begin{bmatrix}8&11&-10\\25&30&-29\\-27&-30&22\end{bmatrix}$.

Реализация в системе GAP

В системе компьютерной алгебры GAP $n\times m$-матрица задается следующим образом: $[[a_{11},a_{12},...,a_{1m}],[a_{21},a_{22},...,a_{2m}],...,[a_{n1},a_{n2},...,a_{nm}]]$. В этом случае матрица будет выведена в виде списка. Для вывода матрицы в стандартном виде применяется функция Display(...). Например

```
gap> A:=[[1,2,3],[4,5,6],[7,8,9]];
[ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ]
gap> Display(A);
[ [ 1, 2, 3 ],
[ 4, 5, 6 ],
[ 7, 8, 9 ] ]
```

Задание матриц специального вида осуществляется при помощи следующих функций системы GAP:

- IdentityMat(m) возвращает единичную матрицу порядка m;
- NullMat(m,n) возвращает нулевую $n \times m$-матрицу;
- DiagonalMat($[a_1, a_2, ..., a_n]$) возвращает диагональную матрицу с диагональными элементами $a_1, a_2, ..., a_n$;
- RandomMat(m,n [,R]) возвращает случайную $n \times m$-матрицу с элементами из кольца R. Если R не указано, то матрица формируется из целых чисел;
- RandomInvertibleMat(m[,R]) формирует обратимую $m \times m$-матрицу с элементами из кольца R. Если R не указано, то матрица формируется из целых чисел.

Обращение к элементу матрицы A производится с помощью команды $A[i][j]$, где i и j — номера строки и столбца соответственно. Отдельные элементы матрицы можно изменять.

```
gap> A[3][2];
8
gap> A[3][2]:=100;
100
```

```
gap> Display(A);
[ [ 1, 2, 3 ],
[ 4, 5, 6 ],
[ 7, 100, 9 ] ]
```

Подматрицы извлекаются или изменяются с помощью фигурных скобок (первая пара скобок указывает выбранные строки, вторая — столбцы):

```
gap> A:=RandomMat(5,9);; Display(A);
[ [ -4, 1, -1, -1, 0, 4, 0, 1, 2 ],
[ -4, 3, -2, -3, 0, 0, 3, -2, 1 ],
[ 0, -4, 3, -2, -2, 1, 1, 1, -3 ],
[ -1, 0, -1, -1, 1, -2, 2, 2, 1 ],
[ -1, 0, 1, -2, 5, 0, 0, 3, 2 ] ]
gap> sA:=A{[1,2]}{[3..5]};;Display(sA);
[ [ -1, -1, 0 ],
[ -2, -3, 0 ] ]
gap> sA{[1,2]}{[3]}:=[[99],[100]];; Display(sA);
[ [ -1, -1, 99 ],
[ -2, -3, 100 ] ]
```

Операции над матрицами в системе GAP:

• A+B — сумма матриц A и B. Если к матрице прибавить скаляр, то на скаляр увеличится каждый элемент матрицы;

• A-B — разность матриц A и B. Если из матрицы вычесть скаляр, то на скаляр уменьшится каждый элемент матрицы;

• A*B — произведение матриц A и B. Матрицу также можно умножать на скаляр и вектор. Следует отметить, что в GAP умножение матриц обобщается, поэтому оно возможно также при несоответствии ;

• A ^ n — возведение матрицы A в степень n;

• Inverse(A) возвращает матрицу, обратную к матрице A. Обратную к A матрицу также можно вычислить с помощью команды A^{-1} или 1/A;

• TransposedMat(A) возвращает матрицу, транспонированую к матрице A.

Пример 4.1.3. Вычислите значение многочлена $A^2 - 2A + 3E_2$, если

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}.$$

```
gap>A:=[[1,1,-1],[2,1,0],[-1,0,1]];; Display(A);
[ [ 1, 1, -1 ],
[ 2, 1, 0 ],
[ -1, 0, 1 ] ]
gap> A^2-2*A+3*IdentityMat(3);
[ [ 5, 0, 0 ], [ 0, 4, -2 ], [ 0, -1, 3 ] ]
gap> Display(last);
[ [ 5, 0, 0 ],
[ 0, 4, -2 ],
[ 0, -1, 3 ] ]
```

Попытка произвести какие-либо действия над транспонированной с помощью функции TransposedMat(A) матрицей приведет к ошибке:

```
gap> B:=TransposedMat(A);;
gap> Display(B);
[ [ 1, 4, 7 ],
[ 2, 5, 100 ],
[ 3, 6, 9 ] ]
gap> B[1][2]:=111;
Lists Assignment: <list> must be a mutable list
not in any function
Entering break read-eval-print loop ...
```

```
you can 'quit;' to quit to outer loop, or
you can 'return;' and ignore the assignment to continue
```

Для того чтобы транспонированная матрица была доступна для изменений, необходимо воспользоваться следующей функцией:

• TransposedMatDestructive(A) возвращает матрицу, транспонированную к матрице A. Полученная матрица доступна для изменений.

```
gap> C:=TransposedMatDestructive(A);;
gap> Display(C);
[ [ 1, 4, 7 ],
[ 2, 5, 100 ],
[ 3, 6, 9 ] ]
gap> C[1][3]:=101;
101
gap> Display(C);
[ [ 1, 4, 101 ],
[ 2, 5, 100 ],
[ 3, 6, 9 ] ]
```

Следующие функции предназначены для проверки определенных свойств матриц:

• IsDiagonalMat(A) возвращает значение true, если матрица A является диагональной. В противном случае возвращает значение false;

• IsUppertiangularMat(A) возвращает значение true, если матрица A является верхней треугольной. В противном случае возвращает значение false;

• IsLowerTriangularMat(A) возвращает значение true, если матрица A является нижней треугольной. В противном случае возвращает значение false.

```
gap> A:=RandomMat(5,4);;Display(A);
[ [ 1, -1, 3, -1 ],
[ -4, -1, -1, 4 ],
[ -1, 2, -4, 1 ],
[ -1, -1, 0, 4 ],
[ 0, 1, 2, -4 ] ]
gap> IsDiagonalMat(A);
false
```

Функции для преобразования матриц:

• TriangulizeMat(A) преобразует матрицу A в верхнюю треугольную матрицу;

• DiagonalizeMat(ring, A) преобразует матрицу A, рассматриваемую над кольцом $ring$, в диагональную.

```
gap> A:=RandomMat(7,5);;Display(A);
[ [ -2, 1, 1, 1, -3 ],
[ -1, 0, -1, -1, 1 ],
```

```
[ -2, 2, 2, 1, -1 ],
[ 0, 1, -2, 5, 0 ],
[ 0, 3, 2, -3, 3 ],
[ 2, 1, 0, 0, -2 ],
[ -3, -2, 1, -4, -3 ] ]
gap> TriangulizeMat(A);Display(A);
[ [ 1, 0, 0, 0, 0 ],
[ 0, 1, 0, 0, 0 ],
[ 0, 0, 1, 0, 0 ],
[ 0, 0, 0, 1, 0 ],
[ 0, 0, 0, 0, 1 ],
[ 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0 ] ]
gap> IsDiagonalMat(A);
true
```

4.2 Определитель матрицы

Рассмотрим квадратную матрицу A порядка n над полем $\mathbb{P}$

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & & \dots & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}.$$

Определение 4.2.1. *Определителем* (или *детерминантом*) квадратной матрицы A называется сумма $n!$ слагаемых, каждое из которых есть произведение n сомножителей, взятых по одному из каждой строки и каждого столбца матрицы A со знаком $sgn\ \tau$, где τ — подстановка из симметрической группы S_n:

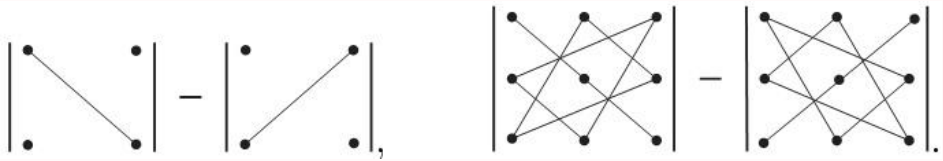
$$sgn\ \tau = \begin{cases} -1, \text{если подстановка } \tau \text{ нечетная}, \\ 1, \text{если подстановка } \tau \text{ четная}. \end{cases}$$

Определитель матрицы A обозначается через $det\ A$.

Другими словами, определитель матрицы A — это элемент поля $\mathbb{P}$, который вычисляется по следующей формуле:

$$detA = \sum_{\tau \in S_n} sgn\ \tau a_{1\tau(1)} a_{2\tau(2)} \dots a_{n\tau(n)}.$$

Формулы вычисления определителей второго и третьего порядка можно представить с помощью следующих рисунков:



Теорема 4.2.1.

1. Если матрица имеет нулевую строку или нулевой столбец, то ее определитель равен нулю.

2. $detA = detA^T$, т.е. при транспонировании матрицы ее определитель не меняется.

3. Если матрица B получается из матрицы A в результате перестановки двух строк, то $detB = -detA$.

4. Определитель матрицы, у которой есть пропорциональные строками, равен нулю.

5. Определитель треугольной матрицы равен произведению диагональных элементов.

6. Определитель диагональной матрицы равен произведению диагональных элементов. В частности, определитель единичной матрицы равен единице.

Теорема 4.2.2. Если A и B — квадратные матрицы одного порядка, то $detAB = detA \cdot detB$.

t **Определение 4.2.2.** Матрица называется *невырожденной*, если ее

определитель отличен от нулевого элемента поля $\mathbb{P}$.

Теорема 4.2.3. Обратимая матрица A является невырожденной и $det(A^{-1}) = (detA)^{-1}$.

Определение 4.2.3. Пусть $A = [a_{ij}]$ — квадратная матрица порядка n. *Минором элемента* a_{kl} называется определитель матрицы, полученной из матрицы A путем вычеркивания k-й строки и l-го столбца. Минор элемента a_{kl} обозначается через M_{kl}.

Алгебраическим дополнением элемента a_{kl} называется элемент $(-1)^{k+l}M_{kl}$ и обозначается через A_{kl}.

Теорема 4.2.4(Теорема Лапласа). Определитель квадратной матрицы $A = [a_{ij}]$ равен сумме произведений элементов какой-либо строки (столбца) на их алгебраические дополнения, т.е.

$$detA = a_{k1}A_{k1} + a_{k2}A_{k2} + ... + a_{kn}A_{kn} = a_{1l}A_{1l} + a_{2l}A_{2l} + ... + a_{nl}A_{nl}.$$

Пример 4.2.1. Вычислите определитель

$$\begin{vmatrix} 2 & 1 & -1 \\ 1 & -1 & 1 \\ -2 & 1 & 3 \end{vmatrix}$$

Применим теорему 4.2.4 к элементам первой строки

$$\begin{vmatrix} 2 & 1 & -1 \\ 1 & -1 & 1 \\ -2 & 1 & 3 \end{vmatrix} = 2 \cdot (-1)^{1+1} \cdot \begin{vmatrix} -1 & 1 \\ 1 & 3 \end{vmatrix} + 1 \cdot (-1)^{1+2} \cdot \begin{vmatrix} 1 & 1 \\ -2 & 3 \end{vmatrix} +$$

$$+(-1)\cdot(-1)^{1+3}\cdot\begin{vmatrix} 1 & -1 \\ -2 & 1 \end{vmatrix} = 2\cdot1\cdot(-3-1)+1\cdot(-1)\cdot(3+2)-1\cdot1\cdot(1-2) =$$

$$= -8-5+1 = -12.$$

Ответ: –12.

Для каждого элемента a_{ij} квадратной матрицы $A = [a_{ij}]$ вычислим алгебраическое дополнение A_{ij} и построим матрицу

$$A' = \begin{bmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ & & \dots & \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{bmatrix},$$

которую назовем *присоединенной* (или взаимной) матрицей к матрице A.

Теорема 4.2.5. Если матрица A невырожденна, то она обратима и $A^{-1} = \frac{1}{detA}A'$.

Итак, для невырожденной матрицы A формула обратной матрицы имеет вид

$$A^{-1} = \frac{1}{detA}A' = \frac{1}{detA}\begin{bmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ & & \dots & \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{bmatrix}.$$

Пример 4.2.2. Найдите обратную к матрице

$$A = \begin{bmatrix} 2 & -1 & 2 \\ 1 & 0 & 1 \\ -1 & 2 & -4 \end{bmatrix}.$$

Вычислим определитель матрицы A, а затем найдем присоединенную матрицу.

$$detA = \begin{vmatrix} 2 & -1 & 2 \\ 1 & 0 & 1 \\ -1 & 2 & -4 \end{vmatrix} = (0+1+4) - (0+4+4) = -3.$$

$$A_{11} = (-1)^{1+1} \cdot \begin{vmatrix} 0 & 1 \\ 2 & -4 \end{vmatrix} = -2, A_{12} = (-1)^{1+2} \cdot \begin{vmatrix} 1 & 1 \\ -1 & -4 \end{vmatrix} = 3,$$

$$A_{13} = (-1)^{1+3} \cdot \begin{vmatrix} 1 & 0 \\ -1 & 2 \end{vmatrix} = 2, A_{21} = (-1)^{2+1} \cdot \begin{vmatrix} -1 & 2 \\ 2 & -4 \end{vmatrix} = 0,$$

$$A_{22} = (-1)^{2+2} \cdot \begin{vmatrix} 2 & 2 \\ -1 & -4 \end{vmatrix} = -6, A_{23} = (-1)^{2+3} \cdot \begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} = -3,$$

$$A_{31} = (-1)^{3+1} \cdot \begin{vmatrix} -1 & 2 \\ 0 & 1 \end{vmatrix} = -1, A_{32} = (-1)^{3+2} \cdot \begin{vmatrix} 2 & 2 \\ 1 & 1 \end{vmatrix} = 0,$$

$$A_{33} = (-1)^{3+3} \cdot \begin{vmatrix} 2 & -1 \\ 1 & 0 \end{vmatrix} = 1.$$

$$A' = \begin{bmatrix} -2 & 0 & -1 \\ 3 & -6 & 0 \\ 2 & -3 & 1 \end{bmatrix}, A^{-1} = \frac{1}{3}\begin{bmatrix} -2 & 0 & -1 \\ 3 & -6 & 0 \\ 2 & -3 & 1 \end{bmatrix}.$$

Ответ: $A^{-1} = \begin{bmatrix} \frac{2}{3} & 0 & \frac{1}{3} \\ -1 & 2 & 0 \\ -\frac{2}{3} & 1 & -\frac{1}{3} \end{bmatrix}$.

Пусть $B = [b_{ij}]$ — $k \times l$-матрица. Зафиксируем натуральное число r, не превосходящее k и l. Выделим в матрице r строк с номерами $i_1 < i_2 < ... < i_r$ и r столбцов с номерами $j_1 < j_2 < ... < j_r$. матрицы A, стоящие на пересечении отмеченных строк и столбцов, образуют квадратную матрицу

$$\begin{bmatrix} b_{i_1 j_1} & b_{i_1 j_2} & ... & b_{i_1 j_r} \\ b_{i_2 j_1} & b_{i_2 j_2} & ... & b_{i_2 j_r} \\ & & ... & \\ b_{i_r j_1} & b_{i_r j_2} & ... & b_{i_r j_r} \end{bmatrix}$$

порядка r. Определитель этой матрицы называется *минором r-го порядка матрицы A*. Для каждого $r < \min\{k, l\}$ можно составить несколько миноров r-го порядка.

Определение 4.2.3. *Рангом* матрицы B называется такое натуральное число r, что среди миноров r-го порядка матрицы B имеется хотя бы один не равный нулю, а все миноры $(r+1)$-го порядка, если такие можно составить, равны нулю. Ранг B матрицы обозначается через $r(B)$.

Ранг ненулевой матрицы всегда ≥ 1. У нулевой матрицы все элементы равны нулю, и ее ранг считают равным 0.

Теорема 4.2.6.

1. При элементарных преобразованиях ранг матрицы не меняется.

2. Ранг ступенчатой матрицы равен числу ее ненулевых строк.

Из теоремы 4.2.6 следует алгоритм вычисления ранга матрицы A:

1) привести матрицу A к ступенчатому виду с помощью элементарных преобразований;

2) число ненулевых строк получившейся ступенчатой матрицы будет рангом матрицы A.

Пример 4.2.8. Вычислите ранг матрицы

$$A = \begin{bmatrix} 1 & -1 & 1 & -4 & 2 \\ 2 & 2 & 3 & 1 & 0 \\ 0 & -1 & 3 & 1 & 3 \\ 1 & 0 & -2 & -5 & -1 \end{bmatrix}.$$

С помощью элементарных преобразований приведем матрицу A к ступенчатому виду:

$$A = \begin{bmatrix} 1 & -1 & 1 & -4 & 2 \\ 2 & 2 & 3 & 1 & 0 \\ 0 & -1 & 3 & 1 & 3 \\ 1 & 0 & -2 & -5 & -1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} 1 & -1 & 1 & -4 & 2 \\ 0 & 4 & 1 & 9 & -4 \\ 0 & -1 & 3 & 1 & 3 \\ 0 & 1 & -3 & -1 & -3 \end{bmatrix} \Leftrightarrow$$

$$\Leftrightarrow \begin{bmatrix} 1 & -1 & 1 & -4 & 2 \\ 0 & 1 & \frac{1}{4} & \frac{9}{4} & -1 \\ 0 & -1 & 3 & 1 & 3 \\ 0 & 1 & -3 & -1 & -3 \end{bmatrix} \Leftrightarrow \begin{bmatrix} 1 & -1 & 1 & -4 & 2 \\ 0 & 1 & \frac{1}{4} & \frac{9}{4} & -1 \\ 0 & 0 & \frac{13}{4} & \frac{13}{4} & 2 \\ 0 & 0 & -\frac{13}{4} & -\frac{13}{4} & -2 \end{bmatrix} \Leftrightarrow$$

$$\Leftrightarrow \begin{bmatrix} 1 & -1 & 1 & -4 & 2 \\ 0 & 1 & \frac{1}{4} & \frac{9}{4} & -1 \\ 0 & 0 & 1 & 1 & \frac{8}{13} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Следовательно, $r(A) = 3$.

Ответ: $r(A) = 3$.

Реализация в системе GAP

Для определения ранга матрицы и вычисления ее определителя используются следующие функции:

- RankMat(A) возвращает ранг матрицы A;
- DeterminantMat(A) вычисляет определитель матрицы A.

Пример 4.2.9. Вычислите ранг матрицы

$$A = \begin{bmatrix} 1 & -2 & 1 & -4 & 2 \\ 2 & -4 & 3 & 1 & 0 \\ 0 & 1 & -1 & 3 & 1 \\ 4 & -7 & 4 & -4 & 5 \end{bmatrix}.$$

```
gap> A:=[[1,-2,1,-4,2],[2,-4,3,1,0],[0,1,-1,3,1],[4,-7,4,-4,5]];;   gap>
 Display(A);
[ [ 1, -2, 1, -4, 2 ],
[ 2, -4, 3, 1, 0 ],
[ 0, 1, -1, 3, 1 ],
[ 4, -7, 4, -4, 5 ] ]
gap> Rank(A);
3
```

Пример 4.2.10. Вычислите определитель

$$\begin{vmatrix} 1 & 2 & -1 \\ 2 & -1 & -2 \\ -2 & -1 & 1 \end{vmatrix}.$$

```
gap> B:=[[1,2,-1],[2,-1,-2],[-2,-1,1]];;
gap> Display(B);
[ [ 1, 2, -1 ],
[ 2, -1, -2 ],
[ -2, -1, 1 ] ]
gap> Determinant(B);
5
```

Пример 4.2.11. Разработайте функцию для вычисления алгебраических дополнений.

```
AlgDop:=function(A,n,i,j)
local Rez,X,D,k1,k2;
Rez:=(-1)^(i+j);
X:=RandomMat(n-1,n-1);
for k1 in [1..n] do
for k2 in [1..n] do
if k1<i then
if k2<j then
X[k1][k2]:=A[k1][k2];
else
if k2>j then
X[k1][k2-1]:=A[k1][k2];
fi;
fi;
else
if k1>i then
if k2<j then
X[k1-1][k2]:=A[k1][k2];
else
```

```
if k2>j then
X[k1-1][k2-1]:=A[k1][k2];
fi;
fi;
fi;
fi;
od;
od;
D:=DeterminantMat(X);
Rez:=Rez*D;
return Rez;
end;
gap> Read("C:/gap4r7/bin/AlgDop.g");
gap> A:=RandomMat(7,7);; Display(A);
[ [ -1,  0,  0, -1, -1,  1,  0 ],
  [  1, -5, -4,  3, -6, -3,  3 ],
  [ -2,  0,  1, -2, -1, -6, -1 ],
  [ -1, -1,  4,  2,  2,  1,  1 ],
  [ -1, -4,  4, -1,  2,  0, -1 ],
  [  3,  3, -1, -1,  3,  1,  0 ],
  [  1,  1, -2,  3, -1,  3, -3 ] ]
```

```
gap> AlgDop(A,7,1,2);
-3380
gap> AlgDop(A,7,6,1);
-7485
gap> AlgDop(A,7,5,7);
926
gap> AlgDop(A,7,3,2);
-1439
gap> B:=RandomMat(3,3);; Display(B);
[ [ -3, -1, -2 ],
[ -3, 1, 0 ],
[ -1, -2, -1 ] ]
gap> AlgDop(B,3,3,2);
6
gap> AlgDop(B,3,1,3);
7
gap> AlgDop(B,3,2,3);
-5
```

Пример 4.2.12. Разработайте функцию для построения присоединенной (взаимной) матрицы к заданной матрице A. Постройте присоединенную матрицу к матрице

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 4 & 1 \\ -1 & 5 & 2 \end{bmatrix}.$$

```
MutualMat:=function(A,n)
local Rez,i,j;
Rez:=RandomMat(n,n);
for i in [1..n] do
for j in [1..n] do
Rez[j][i]:=AlgDop(A,n,i,j);
od;
od;
return Rez;
end;
```

Теперь с помощью разработаной функции MutualMat построим присоединенную к A матрицу:

```
gap> A:=[[1,2,1],[1,4,1],[-1,5,2]];; Display(A);
[ [ 1, 2, 1 ],
[ 1, 4, 1 ],
[ -1, 5, 2 ] ]
gap> Read("C:/gap4r7/bin/AlgDop.g");
gap> Read("C:/gap4r7/bin/MutualMat.g");
gap> MutualMat(A,3);; Display(last);
[ [ 3, 1, -2 ],
[ -3, 3, 0 ],
[ 9, -7, 2 ] ]
```

4.3 Системы линейных уравнений (СЛУ)

Определение 4.3.1. *Системой k линейных уравнений* (СЛУ) с l неизвестными $x_1, x_2, ..., x_l$ над полем $\mathbb{P}$ называется совокупность

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1l}x_l = b_1, \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2l}x_l = b_2, \\ ... \\ a_{k1}x_1 + a_{k2}x_2 + ... + a_{kl}x_l = b_k. \end{cases} \tag{4.3.1}$$

Здесь a_{ij} — элементы поля $\mathbb{P}$, которые называются *коэффициентами*, а элементы b_i также принадлежат $\mathbb{P}$ и называются *свободными коэффи-*

центами системы.

Система 4.3.1, имеющая хотя бы одно решение, называется *совместной*, а система, не имеющая ни одного решения, — *несовместной*. Если СЛУ имеет единственное решение, то ее называют *определенной*, если решений больше одного, то *неопределенной*.

Из коэффициентов при неизвестных в системе 4.3.1 составим $k \times l$-матрицу

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1l} \\ a_{21} & a_{22} & \dots & a_{2l} \\ \dots & & & \\ a_{k1} & a_{k2} & \dots & a_{kl} \end{bmatrix},$$

которая называется *матрицей системы* уравнений 4.3.1. Если к ней дописать столбец свободных коэффициентов, то получим *расширенную матрицу системы* уравнений 4.3.1:

$$A = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1l} & b_1 \\ a_{21} & a_{22} & \dots & a_{2l} & b_2 \\ \dots & & & & \dots \\ a_{k1} & a_{k2} & \dots & a_{kl} & b_k \end{array}\right].$$

Форму 4.3.1 записи СЛУ называют *координатной*, а форму $AX = B$, где A — матрица системы 4.3.1, а

$$B = \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_k \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_k \end{bmatrix}$$

называют *матричной*.

4.3.1 Метод Гаусса решения СЛУ

Метод Гаусса (или метод последовательного исключения неизвестных), основан на приведении системы линейных уравнений к ступенчатому виду. Процесс решения по методу Гаусса состоит из двух этапов.

На первом этапе осуществляется так называемый прямой ход, когда путем элементарных преобразований над строками систему приводят к ступенчатой. Под *элементарными преобразованиями системы линейных уравнений* понимаются следующие преобразования:

1) умножение какого-либо уравнения на ненулевой элемент поля $\mathbb{P}$;

2) прибавление к одному уравнению другого уравнения, умноженного на произвольный элемент поля $\mathbb{P}$.

Заметим, что если в процессе элементарных преобразований получаем уравнение

$$0x_1 + 0x_2 + ...0x_l = 0,$$

то такое уравнение можно убрать из системы, т.к. ему удовлетворяют все элементы поля $\mathbb{P}$.

Если в процессе элементарных преобразований приходим к уравнению

$$0x_1 + 0x_2 + ...0x_l = b, b \neq 0,$$

то система с таким уравнением будет несовместной, т.к. ему не удовлетворяет ни один набор элементов поля $\mathbb{P}$.

На втором этапе осуществляется обратный ход метода Гаусса, который заключается в последовательном выражении переменных. Эта процедура начинается с последнего уравнения, из которого выражают соответствующую базисную переменную (первая переменная слева, коэффициент при которой отличен от нуля) и подставляют в предыдущие уравнения, и так далее, поднимаясь по «ступенькам» наверх.

Т.к. каждому элементарному преобразованию системы соответсвует элементарное преобразование расширенной матрицы этой системы, то вместо системы можно оперировать с расширенной матрицей.

Теорема 4.3.1 (Теорема Кронекера-Капелли). Система 4.3.1 совместна тогда и только тогда, когда ранг ее основной матрицы равен рангу ее расширенной матрицы. Причем совместная система определена, если ее ранг равен числу переменных данной системы.

Метод Гаусса идеально подходит для решения систем уравнений, у которых матрица системы не является квадратной (чего не скажешь про метод Крамера и матричный метод), т.е. метод Гаусса — наиболее

универсальный метод для нахождения решения любой системы линейных уравнений. Он работает в случае, когда система имеет бесконечно много решений или несовместна.

Для иллюстрации метода Гаусса рассмотрим несколько примеров.

Пример 4.3.1. Решите систему линейных уравнений

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 4, \\ 5x_1 + 6x_2 + 7x_3 = 8, \\ 9x_1 + 18x_2 + 27x_3 = 2016. \end{cases}$$

Для решения системы линейных уравнений методом Гаусса запишем расширенную матрицу системы:

$$A = \left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 18 & 27 & 2016 \end{array}\right].$$

С помощью элементарных преобразований приведем расширенную матрицу к ступенчатому виду:

$$A = \left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & -12 \\ 0 & 0 & 0 & 1980 \end{array}\right].$$

Последнее уравнение имеет вид $0 = const$. Поэтому система несовместна и, следовательно, не имеет решений.

Ответ: нет решений.

Пример 4.3.2. Решите систему линейных уравнений

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 4, \\ 5x_1 + 6x_2 + 7x_3 = 8, \\ 9x_1 + 10x_2 + 12x_3 = 12. \end{cases}$$

Составим расширенную матрицу системы:

$$A = \left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 12 & 12 \end{array}\right].$$

С помощью элементарных преобразований приведем расширенную матрицу системы к ступенчатому виду. Получим

$$A = \left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & -12 \\ 0 & 0 & 1 & 0 \end{array}\right].$$

Таким образом, начальная система равносильна следующей ступенчатой системе:

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 4, \\ x_2 + 2x_3 = 3, \\ x_3 = 0. \end{cases}$$

Система линейных уравнений совместная и определенная. Единственное решение системы $x_1 = -2, x_2 = 3, x_3 = 0$.

Ответ: $x_1 = -2, x_2 = 3, x_3 = 0$.

Пример 4.3.3. Решите систему линейных уравнений

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 4, \\ 5x_1 + 6x_2 + 7x_3 = 8, \\ 9x_1 + 18x_2 + 27x_3 = 36. \end{cases}$$

Составим расширенную матрицу системы:

$$A = \left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 18 & 27 & 36 \end{array}\right].$$

С помощью элементарных преобразований приведем расширенную матрицу системы к ступенчатому виду. Получим

$$A = \left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 \end{array}\right].$$

Последнюю строку матрицы можно убрать, т.к. она нулевая:

$$A = \left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 \end{array}\right].$$

Система линейных уравнений совместная и неопределенная. Получим систему

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 4, \\ x_2 + 2x_3 = 3, \end{cases}$$

где x_1, x_2 — базисные переменные, x_3 — свободная переменная.

Придадим неизвестной x_3 произвольное значение из поля действительных чисел: $x_3 = \lambda$. Из второго уравнения полученной системы выразим переменную x_2 через x_3: $x_2 = 3-2x_3 = 3-2\lambda$. Подставим полученное выражение для x_2 в первое уравнение системы и выразим переменную x_1 через переменную x_3: $x_1 = 4-2x_2-3x_3 = 4-2(3-2\lambda)-3\lambda = -2+\lambda$.

Т.к. последняя система равносильна исходной, то формулы

$$\begin{cases} x_1 = -2 + \lambda, \\ x_2 = 3 - 2\lambda, \\ x_3 = \lambda \end{cases}$$

при произвольном λ дают нам решения заданной системы.

Ответ: $x_1 = -2 + \lambda$, $x_2 = 3 - 2\lambda$, $x_3 = \lambda$, где λ имеет произвольное действительное значение.

Реализация в системе GAP

Для решения систем линейных уравнений в системе GAP существует две функции:

• NullspaceMat(A) возвращает базис пространства решений системы $X \cdot A = 0$;

• SolutionMat(A, B) возвращает вектор X, который является решением уравнения $X \cdot A = B$. Возвращает $fail$, если такого вектора не

существует.

Пример 4.3.4. Решите систему линейных уравнений

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 4, \\ 5x_1 + 6x_2 + 7x_3 = 8, \\ 9x_1 + 18x_2 + 27x_3 = 2016. \end{cases}$$

Выполним проверку совместности СЛУ по теореме Кронекера-Капелли:

```
gap> A:=[[1,2,3],[5,6,7],[9,18,27]];;
gap> Ar:=[[1,2,3,4],[5,6,7,8],[9,18,27,2016]];;
gap> Rank(Ar)=Rank(A);
false
```

Т.к. ранг матрицы A не равен рангу расширенной матрицы Ar, система является несовместной. Если мы к этой системе применим функцию SolutionMat(A, B), то GAP выдаст сообщение fail.

```
gap> At:=TransposedMatDestructive(A);;
gap> B:=[4,8,2016];;
gap> SolutionMat(At,B);
fail
```

Пример 4.3.5. Решите систему линейных уравнений

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 4, \\ 5x_1 + 6x_2 + 7x_3 = 8, \\ 9x_1 + 10x_2 + 12x_3 = 12. \end{cases}$$

Выполним проверку совместности СЛУ по теореме Кронекера-Капелли:

```
gap> A:=[[1,2,3],[5,6,7],[9,10,12]];; Display(A);
[ [ 1, 2, 3 ],
[ 5, 6, 7 ],
[ 9, 10, 12 ] ]
gap> Ar:=[[1,2,3,4],[5,6,7,8],[9,10,12,12]];; Display(Ar);
[ [ 1, 2, 3, 4 ],
[ 5, 6, 7, 8 ],
[ 9, 10, 12, 12 ] ]
gap> Rank(A);
3
gap> Rank(Ar)=Rank(A);
true
```

Следовательно, система является совместной. Т.к. ранг матрицы равен количеству переменных, то система является определенной. Применение функции SolutionMat(A, B) выдаст вектор, являющийся единственным решением системы.

```
gap> At:=TransposedMatDestructive(A);; Display(At);
[ [ 1, 5, 9 ],
[ 2, 6, 10 ],
[ 3, 7, 12 ] ]
gap> B:=[4,8,12];
[ 4, 8, 12 ]
gap> SolutionMat(At,B);
[ -2, 3, 0 ]
```

Пример 4.3.6. Решите систему линейных уравнений

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 4, \\ 5x_1 + 6x_2 + 7x_3 = 8, \\ 9x_1 + 18x_2 + 27x_3 = 36. \end{cases}$$

Выполним проверку совместности СЛУ по теореме Кронекера-Капелли:

```
gap> A:=[[1,2,3],[5,6,7],[9,18,27]];; Display(A);
[ [ 1, 2, 3 ],
[ 5, 6, 7 ],
[ 9, 18, 27 ] ]
gap> Ar:=[[1,2,3,4],[5,6,7,8],[9,18,27,36]];; Display(Ar);
```

```
[ [ 1, 2, 3, 4 ],
[ 5, 6, 7, 8 ],
[ 9, 18, 27, 36 ] ]
gap> Rank(A);
2
gap> Rank(Ar)=Rank(A);
true
```

Следовательно, система является совместной. Т.к. ранг матрицы меньше количества переменных, то система является неопределенной. Если мы применим к системе 4.3.1 функцию SolutionMat(A, B), то получим только одно решение.

```
gap> At:=TransposedMatDestructive(A);; Display(At);
[ [ 1, 5, 9 ],
[ 2, 6, 18 ],
[ 3, 7, 27 ] ]
gap> B:=[4,8,36];
[ 4, 8, 36 ]
gap> SolutionMat(At,B);
[ -2, 3, 0 ]
```

Для получения общего решения системы вида $XA = B$ в случае

неопределенности построим базис пространства решений системы $XA = 0$.

```
gap> NullspaceMat(At);
[ [ 1, -2, 1 ] ]
```

Тогда решением будет являться и следующий вектор:

```
gap> M:=SolutionMat(At,B)+5/2*NullspaceMat(At)[1];
[ 1/2, -2, 5/2 ]
```

Выполним проверку:

```
gap> M*At=B;
true
```

Изменяя коэффициент при NullspaceMat(A)[1], будем получать различные решения нашей системы.

Пример 4.3.7. Разработайте функцию решения СЛУ вида $AX = B$, где A — квадратная матрица порядка n.

```
GAUSS:=function(A,B,n)
local RankA, RankAR, AT, AR, Ar, d, R, BazisR;
RankA:=Rank(A);
d:=DeterminantMat(A);
AT:=TransposedMatDestructive(A);
Ar:=AT;
```

```
Add(Ar,B);
AR:=TransposedMatDestructive(Ar);
RankAR:=Rank(AR);
A:=AR{[1..n]}{[1..n]};
AT:=TransposedMatDestructive(A);
if RankAR<>RankA then
R:="Net resheniy";
Print(R,"\n");
else
if RankAR=RankA and d<>0 then
R:=SolutionMat(AT,B);
return R;
else
R:=SolutionMat(AT,B);
BazisR:=NullspaceMat(AT)[1];
Print(R,"+K * " ,BazisR, " , gde K - proizvol'noe deystvitel'noe
chislo" , "\n");
fi;
fi;
end;
```

Теперь, используя разработанную функцию GAUSS, решим СЛУ из

примеров 4.3.4, 4.3.5 и 4.3.6. Для этого запустим функцию GAUSS:

```
gap> Read("C:/gap4r7/bin/GAUSS.g");
```

Решение СЛУ из примера 4.3.4:

```
gap> A1:=[[1,2,3],[5,6,7],[9,18,27]];;
gap> B1:=[4,8,2016];;
gap> n:=3;;
```

```
gap> GAUSS(A1,B1,n);
Net resheniy
```

Решение СЛУ из примера 4.3.5:

```
gap> A2:=[[1,2,3],[5,6,7],[9,10,12]];;
gap> B2:=[4,8,12];;
gap> n:=3;;
gap> GAUSS(A2,B2,n);
[ -2, 3, 0 ]
```

Решение СЛУ из примера 4.3.6:

```
gap> A3:=[[1,2,3],[5,6,7],[9,18,27]];;
gap> B3:=[4,8,36];;
gap> n:=3;;
gap> GAUSS(A3,B3,n);
[ -2, 3, 0 ]+K*[ 1, -2, 1 ], gde K - proizvol'noe deystvitel'noe chislo
```

4.3.2 Метод Крамера решения СЛУ

Если число уравнений равно числу неизвестных ($k = l$) и определитель матрицы системы отличен от нуля, то система называется *крамеровской*. Крамеровская система имеет единственное решение, которое определяется по формулам

$$x_i = \frac{\Delta_i}{\Delta}, i = 1, 2, ..., k,$$

где $\Delta = detA$ — определитель матрицы системы, а Δ_i — определитель матрицы, которая получена из A заменой i-го столбца на столбец свободных членов.

Пример 4.3.8. Решите систему линейных уравнений

$$\begin{cases} 2x_1 + 5x_2 + 4x_3 = 30, \\ x_1 + 3x_2 + 2x_3 = 150, \\ 2x_1 + 10x_2 + 9x_3 = 110. \end{cases}$$

Вычислим определитель матрицы системы

$$\Delta = detA = \begin{vmatrix} 2 & 5 & 4 \\ 1 & 3 & 2 \\ 2 & 10 & 9 \end{vmatrix} = \begin{vmatrix} 0 & -1 & 0 \\ 1 & 3 & 2 \\ 0 & 4 & 5 \end{vmatrix} = \begin{vmatrix} -1 & 0 \\ 4 & 5 \end{vmatrix} = 5.$$

Вычислим определители $\Delta_i, i = 1, 2, 3$, где Δ_i — определитель матрицы, у которой в i-м столбце стоят свободные коэффициенты 30, 150, 110, а остальные столбцы, как у матрицы системы.

$$\Delta_1 = \begin{vmatrix} 30 & 5 & 4 \\ 150 & 3 & 2 \\ 110 & 10 & 9 \end{vmatrix} = \begin{vmatrix} 30 & 5 & 4 \\ 150 & 3 & 2 \\ -10 & -190 & -7 \end{vmatrix} = \begin{vmatrix} 0 & -25 & -17 \\ 0 & -147 & -103 \\ -10 & -10 & -7 \end{vmatrix} =$$

$$= -10 \cdot (-1)^{1+3} \cdot \begin{vmatrix} -25 & -17 \\ -147 & -103 \end{vmatrix} = -10 \cdot (-1)^{1+3} \cdot ((-25) \cdot (-103) -$$

$$-(-147) \cdot (-17)) = 760,$$

$$\Delta_2 = \begin{vmatrix} 2 & 30 & 4 \\ 1 & 150 & 2 \\ 2 & 110 & 9 \end{vmatrix} = \begin{vmatrix} 0 & -270 & 0 \\ 1 & 150 & 2 \\ 0 & -190 & 5 \end{vmatrix} = (-1)^{1+2} \cdot \begin{vmatrix} -270 & 0 \\ -190 & -5 \end{vmatrix} =$$

$$= (-1)^{1+2} \cdot (-270) \cdot 5 = 1350,$$

$$\Delta_3 = \begin{vmatrix} 2 & 5 & 30 \\ 1 & 3 & 150 \\ 2 & 10 & 110 \end{vmatrix} = \begin{vmatrix} 0 & -1 & -270 \\ 1 & 3 & 150 \\ 0 & 4 & -190 \end{vmatrix} = (-1)^{1+2} \cdot \begin{vmatrix} -1 & -270 \\ 4 & -190 \end{vmatrix} =$$

$= (-1)^{1+2} \cdot ((-1) \cdot (-190) - 4 \cdot (-270)) = -(190 + 1080) = -1270.$

Находим решение:

$$x_1 = \frac{\Delta_1}{\Delta} = \frac{-760}{5} = -152, x_2 = \frac{\Delta_2}{\Delta} = \frac{1350}{5} = 270,$$

$$x_3 = \frac{\Delta_3}{\Delta} = \frac{-1270}{5} = -254.$$

Реализация в системе GAP

Пример 4.3.9. Составить функцию для решения СЛУ методом Крамера. С помощью полученной функции решить систему линейных уравнений

$$\begin{cases} 2x_1 + 5x_2 + 4x_3 = 30, \\ x_1 + 3x_2 + 2x_3 = 150, \\ 2x_1 + 10x_2 + 9x_3 = 110. \end{cases}$$

Вычислим определитель матрицы системы:

```
gap>A:=[[2,5,4],[1,3,2],[2,10,9]];; Display(A);
[ [ 2, 5, 4 ],
[ 1, 3, 2 ],
[ 2, 10, 9 ] ]
gap> Determinant(A);
5
```

Следовательно, система является крамеровской. Разработаем функцию, решающую СЛУ методом Крамера.

```
KRAMER:=function(A,B,n)
local D, l, i, j, k, X, Dx;
D:=DeterminantMat(A);
if D<>0 then
l:=[];
for k in [1..n] do
X:=NullMat(n,n);
for i in [1..n] do
for j in [1..n] do
X[i][j]:=A[i][j];
od;
od;
for i in [1..n] do
X[i][k]:=B[i];
od;
Dx:=DeterminantMat(X);
l[k]:=Dx/D;
od;
```

```
Print(l,"\n");
else Print("Systema ne yavlyaetsya Kramerovskoy" ,"\n");
fi;
end;
```

Теперь решим систему, воспользовавшись функцией KRAMER:

```
gap> Read("C:/gap4r4/bin/KRAMER.g");
gap> KRAMER(A,B,3);
[ -152, 270, -254 ]
```

4.3.3 Матричный метод решения СЛУ

Рассмотрим систему линейных уравнений 3.4.1 в матричной форме

$$AX = B. \tag{3.4.2}$$

Если матрица A является невырожденной, то для нее существует обратная A^{-1}. Домножим обе части равенства 3.4.2 на A^{-1} слева. Получим формулу для нахождения X:

$$A^{-1}(AX) = A^{-1}B, X = A^{-1}B.$$

Пример 4.3.10. Решите систему линейных уравнений

$$\begin{cases} 3x_1 + 2x_2 - x_3 = 4, \\ 2x_1 - x_2 + 5x_3 = 23, \\ x_1 + 7x_2 - x_3 = 5. \end{cases}$$

Для начала убедимся в том, что определитель матрицы системы линейных уравнений не равен нулю:

$$detA = \begin{vmatrix} 3 & 2 & -1 \\ 2 & -1 & 5 \\ 1 & 7 & -1 \end{vmatrix} = 3 - 14 + 10 - (1 + 105 - 4) = -105.$$

Теперь вычислим алгебраические дополнения для элементов матрицы системы. Они нам понадобятся для нахождения обратной матрицы.

$$A_{11} = (-1)^{1+1} \cdot \begin{vmatrix} -1 & 5 \\ 7 & -1 \end{vmatrix} = -34; A_{12} = (-1)^{1+2} \cdot \begin{vmatrix} 2 & 5 \\ 1 & -1 \end{vmatrix} = 7;$$

$$A_{13} = (-1)^{1+3} \cdot \begin{vmatrix} 2 & -1 \\ 1 & 7 \end{vmatrix} = 15; A_{21} = (-1)^{2+1} \cdot \begin{vmatrix} 2 & -1 \\ 7 & -1 \end{vmatrix} = -5;$$

$$A_{22} = (-1)^{2+2} \cdot \begin{vmatrix} 3 & -1 \\ 1 & -1 \end{vmatrix} = -2; A_{23} = (-1)^{2+3} \cdot \begin{vmatrix} 3 & 2 \\ 1 & 7 \end{vmatrix} = -19;$$

$$A_{31} = (-1)^{3+1} \cdot \begin{vmatrix} 2 & -1 \\ -1 & 5 \end{vmatrix} = 9; A_{32} = (-1)^{3+2} \cdot \begin{vmatrix} 3 & -1 \\ 2 & 5 \end{vmatrix} = -17;$$

$$A_{33} = (-1)^{3+3} \cdot \begin{vmatrix} 3 & 2 \\ 2 & -1 \end{vmatrix} = -7.$$

Находим обратную к A матрицу:

$$A^{-1} = \frac{1}{detA} \begin{bmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \frac{1}{-103} \begin{bmatrix} -34 & -5 & 9 \\ 7 & -2 & -17 \\ 15 & -19 & -7 \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{-34}{-103} & \frac{-5}{-103} & \frac{9}{-103} \\ \frac{7}{-103} & \frac{-2}{-103} & \frac{-17}{-103} \\ \frac{15}{-103} & \frac{-19}{-103} & \frac{-7}{-103} \end{bmatrix}.$$

Тогда

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = A_{-1}B = \begin{bmatrix} \frac{-34}{-103} & \frac{-5}{-103} & \frac{9}{-103} \\ \frac{7}{-103} & \frac{-2}{-103} & \frac{-17}{-103} \\ \frac{15}{-103} & \frac{-19}{-103} & \frac{-7}{-103} \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 23 \\ 5 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}.$$

Ответ: $x_1 = 2, x_2 = 1, x_3 = 4.$

Реализация в системе GAP

Пример 4.3.11. Решите систему линейных уравнений

$$\begin{cases} 3x_1 + 2x_2 - x_3 = 4, \\ 2x_1 - x_2 + 5x_3 = 23, \\ x_1 + 7x_2 - x_3 = 5. \end{cases}$$

Вычислим определитель матрицы системы:

```
gap> A:=[[3,2,-1],[2,-1,5],[1,7,-1]];; Display(A);
[ [ 3, 2, -1 ],
[ 2, -1, 5 ],
[ 1, 7, -1 ] ]
gap> Determinant(A);
-103
```

Т.к. определитель матрицы системы отличен от нуля, то система является Крамеровской. Найдем решение:

```
gap> B:=[4,23,5];
[ 4, 23, 5 ]
gap> A ^ (-1)*B;
[ [ 2 ], [ 1 ], [ 4 ] ]
```

РАЗДЕЛ 5
КОЛЬЦО МНОГОЧЛЕНОВ В СИСТЕМЕ GAP. ВВЕДЕНИЕ В ТЕОРИЮ ГАЛУА

5.1 Кольцо многочленов

Пусть $\mathbb{P}$ — это произвольное поле, 0 и 1 — нулевой и единичный элемент поля $\mathbb{P}$.

Определение 5.1.1. *Многочленом (полиномом)* от переменной x над полем $\mathbb{P}$ называется выражение

$$f(x) = a_0x^0 + a_1x^1 + a_2x^2 + ... + a_nx^n, \tag{5.1.1}$$

где $n \in \mathbb{N}\backslash\{0\}$, $a_i \in \mathbb{P}, i = \overline{0,n}$ называются *коэффициентами* многочлена $f(x)$. Наибольшее число $k \in \mathbb{N} \setminus \{0\}$, такое что $a_k \neq 0 \in \mathbb{P}$ называется *степенью* многочлена $f(x)$ и обозначается через $deg\ f(x) = k$.

Пример 5.1.1. Степень многочлена $f(x) = 9x^0 + 0 \cdot x + 2x^2 + 0 \cdot x^3$ равна 2, т.е. $deg\ f(x) = 2$.

Рассмотрим многочлен $f(x) = a_0x^0$. Если $a_0 \neq 0$, то степень многочлена $f(x)$ равна 0. Если же $a_0 = 0$, то степень многочлена $f(x) = 0 \cdot x^0$ не определена, а сам многочлен называется *нулевым*.

Если $deg\ f(x) = k$, то коэффициент a_k называется *старшим*.

Определение 5.1.2. Многочлен, старший коэффициент которого равен единице $1 \in \mathbb{P}$, называется *нормированным*.

Определение 5.1.3. Два многочлена $f(x)$ и $g(x)$ над полем $\mathbb{P}$ называются *равными алгебраически*, если равны их коэффициенты при одинаковых степенях и они отличаются лишь слагаемыми с нулевыми коэффициентами. Если два многочлена $f(x)$ и $g(x)$ равны, то пишут $f(x) = g(x)$.

Множество всех многочленов от переменной x над полем $\mathbb{P}$ обозначается через $\mathbb{P}[x]$.

Пусть $f(x) = a_0x^0 + a_1x^1 + a_2x^2 + ... + a_nx^n \in \mathbb{P}[x]$ и $g(x) = b_0x^0 + b_1x^1 + b_2x^2 + ... + b_nx^n \in \mathbb{P}[x]$. Предположим, что $n \geq m$.

Определение 5.1.4. *Суммой* многочленов $f(x)$ и $g(x)$ называется многочлен

$$f(x) + g(x) = c_0x^0 + c_1x^1 + c_2x^2 + ... + c_nx^n,$$

где $c_i = a_i + b_i, i = \overline{0, n}$ (здесь имеется ввиду, что если $i > m$, то $b_i = 0$).

Определение 5.1.5. *Произведением* многочленов $f(x)$ и $g(x)$ называется многочлен

$$f(x) \cdot g(x) = d_0x^0 + d_1x^1 + d_2x^2 + ... + d_{n+m}x^{n+m},$$

где $d_i = \sum_{q+p=i} a_pb_q, i = \overline{0, n+m}$.

Пример 5.1.2. Рассмотрим два многочлена: $f(x) = a_0x^0 + a_1x^1 + a_2x^2$ и $g(x) = b_0x^0 + b_1x^1$. Тогда

$$f(x) + g(x) = (a_0 + b_0)x^0 + (a_1 + b_1)x^1 + a_2x^2$$

и

$$f(x) \cdot g(x) = (a_0b_0)x^0 + (a_0b_1 + a_1b_0)x^1 + (a_2b_0 + a_1b_1)x^2 + (a_2b_1)x^3.$$

Приведем свойства степеней многочленов.

1. Если $f(x)$ и $g(x)$ ненулевые многочлены из $\mathbb{P}[x]$, то степень многочлена $f(x)+g(x)$ не превосходит максимума стеменей многочленов $f(x)$ и $g(x)$, т.е.

$$deg\ (f(x) + g(x)) \leq \max\{deg\ f(x), deg\ g(x)\}.$$

2. Если $f(x)$ и $g(x)$ ненулевые многочлены из $\mathbb{P}[x]$, то степень многочлена $f(x)\cdot g(x)$ равна сумме степеней сомножителей, т.е. $deg\ (f(x)\cdot g(x)) \leq deg\ f(x) + deg\ g(x)$.

Теорема 5.1.1. Множество $\mathbb{P}[x]$ со сложением и умножением является кольцом с единицей $1 \cdot x^0$, где 1 — единичный элемент поля $\mathbb{P}$.

Определение 5.1.6. Кольцо $\mathbb{P}[x]$ называется *кольцом многочленов* от одной переменной x над полем $\mathbb{P}$.

Реализация в системе GAP

Многочлен от одной переменной в системе GAP задается следующим

образом. Сначала определяется кольцо коэффициентов, а затем трансцендентный элемент над этим кольцом:

```
gap> Q:=Rationals;
Rationals
gap> x:=Indeterminate(Q);
x_1
```

Только после этого многочлены можно задавать привычным образом:

```
gap> f:=1+2*x+3*x^2+5*x^4;
1+2*x_1+3*x_1^2+5*x_1^4
gap> g:=(x-2)*(x+1/2);
x_1^2-3/2*x_1-1
```

Не смотря на то, что переменная была обозначена как , при выводе она обозначается как x_1. Для того, чтобы этого избежать нужно ее определить следующим образом:

```
gap> x:=Indeterminate(Q,"x");
x
```

Тогда определенные выше многочлены f и g будут выглядеть так:

```
gap> f:=1+2*x+3*x^2+5*x^4;
5*x^4+3*x^2+2*x+1
gap> g:=(x-2)*(x+1/2);
x^2-3/2*x-1
```

Ниже продемонстрированы действия над многочленами — сложение, вычитание, умножение, сравнение:

```
gap> f+g;
5*x^4+4*x^2+1/2*x
gap> f-g;
5*x^4+2*x^2+7/2*x+2
gap> f*g;
5*x^6-15/2*x^5-2*x^4-5/2*x^3-5*x^2-7/2*x-1
gap> f=g;
false
```

Для определения степени многочлена в GAP существует функция DegreeIndeterminate. Например:

```
gap> DegreeIndeterminate(f,x);
4
gap> DegreeIndeterminate(f*g,x);
6
```

Для работы с коэффициентами многочлена в GAP есть две функции:

• PolynomialCoefficientsOfPolynomial — возвращает список коэффициентов многочлена;

• LeadingCoefficient(f); — возвращает коэффициент при старшем члене многочлена f. Например:

```
gap> PolynomialCoefficientsOfPolynomial(f,x);
[ 1, 2, 3, 0, 5 ]
gap> LeadingCoefficient(f)
5
```

Ассоциированность многочленов проверяется с помощью функции IsAssociated:

```
gap> g:=(x-2)*(x+1/2);
x^2 - 3/2 * x - 1
gap> h:=5*x-1;
5*x-1
gap> IsAssociated(g,h);
false
gap> IsAssociated(h,x-1/5);
true
```

5.2 Деление в кольце многочленов

Теория делимости в кольце многочленов $\mathbb{P}[x]$ во многом параллельна теории делимости в кольце целых чисел Z.

Определение 5.2.1. Многочлен $f(x) \in \mathbb{P}[x]$ *делится* в кольце $\mathbb{P}[x]$ на многочлен $\phi(x) \in \mathbb{P}[x]$, $\phi(x) \neq 0$, если существует многочлен $q(x) \in \mathbb{P}[x]$ такой, что $f(x) = \phi(x) \cdot q(x)$. В данном случает многочлен $f(x)$ называется *делимым* (кратным многочлену $\phi(x)$), многочлен $\phi(x)$ — *делителем* многочлена $f(x)$, а $q(x)$ — *частным* при делении $f(x)$ на $\phi(x)$. Если многочлен $f(x)$ делится на многочлен $\phi(x)$, то пишут $f(x) \vdots \phi(x)$.

В теореме 5.2.1 приведены простейший свойства делимости в кольце многочленов $\mathbb{P}[x]$ (аналогичны свойствам делимости в кольце целых чисел $\mathbb{Z}$).

Теорема 5.2.1. Для ненулевых многочленов из $\mathbb{P}[x]$ справедливы следующие свойства:

1) $f(x) \vdots f(x)$;

2) если $f(x) \vdots \phi(x)$ и $\phi(x) \vdots \psi(x)$, то $f(x) \vdots \psi(x)$;

3) если $g(x) \vdots f_1(x)$ и $g(x) \vdots f_2(x)$, то $g(x) \vdots (f_1(x) \cdot \phi_1(x) \pm f_2(x) \cdot \phi_2(x))$ для любых $\phi_1(x)$ и $\phi_2(x)$ из $\mathbb{P}[x]$;

4) если $g(x) \vdots f(x)$, то $a \cdot g(x) \vdots f(x)$ для любого ненулевого элемента a из поля P;

5) если $f(x) \vdots \phi(x)$ и $\phi(x) \vdots f(x)$, то $f(x) = a \cdot \phi(x)$ для любого нену-

левого элемента a из поля $\mathbb{P}$;. Если к тому же старшие коэффициенты многочленов $f(x)$ и $\phi(x)$ равны, то $f(x) = \phi(x)$.

Определение 5.2.2. Многочлены $f(x)$ и $\phi(x)$ из кольца $\mathbb{P}[x]$, которые отличаются только обратимым в $\mathbb{P}$ множителем называются *ассоциированными.*

Заметим, что деление в кольце $\mathbb{P}[x]$ также как и в кольце целых чисел $\mathbb{Z}$ не всегда возможно.

Определение 5.2.3. Многочлен $f(x) \in \mathbb{P}[x]$ *делится с остатком* в кольце $\mathbb{P}[x]$ на многочлен $\phi(x) \in \mathbb{P}[x]$, $\phi(x) \neq 0$, если существуют такие многочлены $q(x) \in P[x]$ и $r(x) \in \mathbb{P}[x]$, что $f(x) = \phi(x) \cdot q(x) + r(x)$ и $deg\ r(x) \leq deg\ \phi(x)$.

Многочлены $q(x)$ и $r(x)$ называются соответсвтенно *неполным частным* и *остатком при делении* $f(x)$ на $\phi(x)$.

Теорема 5.2.2. (О делении с остатком в кольце многочленов). Всякий многочлен $f(x) \in \mathbb{P}[x]$ однозначно делится с остатком в кольце $\mathbb{P}[x]$ на ненулевой многочлен $\phi(x) \in \mathbb{P}[x]$, т.е. существует единственная пара многочленов $f(x)$ и $\phi(x)$ из $\mathbb{P}[x]$, для которых $f(x) = \phi(x) \cdot q(x) + r(x)$ и $deg\ r(x) \leq deg\ \phi(x)$.

Определение 5.2.4. Если многочлен $d(x)$ делит многочлены $f(x)$ и $g(x)$, то $d(x)$ называется *общим делителем* $f(x)$ и $g(x)$. Если, кроме того, $d(x)$ делится на любой другой общий делитель многочленов $f(x)$ и $g(x)$, то $d(x)$ называется *наибольшим общим делителем.*

Замечание. В отличие от наибольшего общего делителя целых чи-

сел, наибольший общий делитель многочленов определятся неоднозначно. Поэтому через НОД$(f(x), g(x))$ обозначается множество всех наибольших общих делителей многочленов $f(x)$ и $g(x)$. Согласно п.5 теоремы 5.2.1 все наибольшие общие делители двух многочленов $f(x)$ и $g(x)$ отличаются друг от друга на ненулевой элемент поля $\mathbb{P}$.

Наибольший общий делитель многочленов находится с помощью алгоритма Евклида, который основан на многократном применении теоремы о делении с остатком (аналогично нахождению НОД целых чисел).

Пусть $f(x)$ и $g(x)$ два ненулевых многочлена над полем $\mathbb{P}$. Тогда можно записать цепочку равенств.

$$
\begin{gathered}
f(x) = g(x)q_1(x) + r_1(x),\ deg\ r_1(x) < deg\ g(x),\\
g(x) = r_1(x)q_2(x) + r_2(x),\ deg\ r_2(x) < deg\ r_1(x),\\
r_1(x) = r_2(x)q_3(x) + r_3(x),\ deg\ r_3(x) < deg\ r_2(x),\\
\dots\\
r_{n-3}(x) = r_{n-2}(x)q_{n-1}(x) + r_{n-1}(x),\ deg\ r_{n-1}(x) < deg\ r_{n-2}(x),\\
r_{n-2}(x) = r_{n-1}(x)q_n(x) + r_n(x),\ deg\ r_n(x) < deg\ r_{n-1}(x),\\
r_{n-1}(x) = r_n(x)q_{n+1}(x)
\end{gathered}
$$

Поскольку степени остатков строго убывают

$$deg\ g(x) > deg\ r_1(x) > deg\ r_2(x) > \dots > deg\ r_{n-1}(x) > deg\ r_n(x),$$

то через конечное число шагов появится остаток равный нулю.

Теорема 5.2.3. Для любых двух ненулевых многочленов $f(x)$ и $g(x)$ последний ненулевой остаток в алгоритме Евквлида для этих многочленов принадлежит множеству НОД$(f(x), g(x))$.

Определение 5.2.5. Многочлены $f_i(x)$ из $\mathbb{P}[x]$ $(i = \overline{1,n})$ называются *взаимно простыми*, если их наибольшим общим делителем является многочлен нулевой степени.

Определение 5.2.6. *Общим кратным* ненулевых многочленов $f_i(x)$ из $\mathbb{P}[x]$ $(i = \overline{1,n})$ называется такой многочлен $k(x) \in P[x]$, который делится на каждый многочлен $f_i(x)$, т.е. $k(x) \vdots f_i(x)$ для $i = \overline{1,n}$.

Определение 5.2.7. *Наименьшим общим кратным (НОК)* ненулевых многочленов $f_i(x)$ из $\mathbb{P}[x]$ $(i = \overline{1,n})$ называется такое их общее кратное, которое является делителем всякого общего кратного этих многочленов.

Реализация в системе GAP

• Функция Quotient(f,g) возвращает частное от деления многочлена f на многочлен g, если такое деление выполнимо, и fail — в противном случае.

```
gap> f:=1+2*x+3*x^2+5*x^4;
5*x^4+3*x^2+2*x+1
gap> g:=(x-2)*(x+1/2);
x^2-3/2*x-1
```

```
gap> Quotient(f,g);
fail
gap> Quotient(g,x-2);
x+1/2
```

• Функции EuclideanQuotient и EuclideanRemainder используются для деления многочлена на многочлен с остатком. Данные функции возвращают неполное частное и остаток соответственно:

```
gap>q:=EuclideanQuotient(f,g);
5*x^2+15/2*x+77/4
gap> r:=EuclideanRemainder(f,g);
307/8*x+81/4
```

Несложно проверить, что это действительно так:

```
gap> f1:=g*q+r;
5*x^4+3*x^2+2*x+1
gap> f=f1;
```

• Функция Gcd возвращает наибольший общий делитель многочленов:

```
gap>Gcd(f,g);
1
```

```
gap> Gcd(g,x-2);
x-2
```

• Функция GcdRepresentation находит линейное представление наибольшего общего делителя:

```
gap>v:=GcdRepresentation(f,g);
[ -614/1649*x+1245/1649,
3070/1649*x^3-1620/1649*x^2+146/97*x-404/1649 ]
```

Проверим, что оно найдено правильно:

```
gap> Gcd(f,g)=f*v[1]+g*v[2];
true
```

• Функция Lcm вычисляет наименьшее общее кратное многочленов:

```
gap>Lcm(f,g);
x^6-3/2*x^5-2/5*x^4-1/2*x^3-x^2-7/10*x-1/5
```

5.3 Неприводимые многочлены. Разложение многочленов на неприводимые множители

Рассмотрим многочлен $f(x) \in \mathbb{P}[x]$, $deg\ f(x) \geq 1$. Делителями многочлена $f(x)$ будут все многочлены нулевой степени и все ассоциированные с ним многочлены $a_0 f(x)$ (a_0 – ненулевой элемент поля $\mathbb{P}$). Если

многочлен $f(x)$ не имеет других делителей в кольце $\mathbb{P}[x]$, то он называется *неприводимым*. Многочлен, который не является неприводимым над полем $\mathbb{P}$, называется *приводимым*.

Замечание. Приводимость многочленов зависит от поля. Например, многочлен $x^2 - 2 = (x - \sqrt{2})(x + \sqrt{2})$ неприводим над $\mathbb{Q}$, но приводим над $\mathbb{R}$.

В следующей теореме приводятся свойства неприводимых многочленов.

Теорема 5.3.1. Многочлены над полем $\mathbb{P}$ обладают следующими свойствами:

1) всякий многочлен первой степени неприводим над полем $\mathbb{P}$;

2) если многочлен $p(x)$ неприводим над полем $\mathbb{P}$, то над этим полем неприводимым будет и любой ассоциированный с $p(x)$ многочлен, т.е. многочлен $af(x)$ также неприводим над полем $\mathbb{P}$ для любого ненулевого элемента a поля $\mathbb{P}$;

3) если многочлен $f(x)$ неприводим над полем $\mathbb{P}$, то для любого многочлена $g(x) \in \mathbb{P}[x]$ либо $f(x)$ делит $g(x)$, либо многочлены $f(x)$ и $g(x)$ взаимно просты;

4) если произведение $f(x)g(x)$ делится на неприводимый многочлен $h(x)$, то либо $f(x)$ делится на $h(x)$, либо $g(x)$ делится на $h(x)$.

Теорема 5.3.2. Всякий многочлен $f(x) \in \mathbb{P}[x]$, $deg\ f(x) \geq 1$ может быть разложен в произведение неприводимых над полем $\mathbb{P}$ многочленов.

Если имеются два таких разложения

$$f(x) = \phi_1(x) \cdot \phi_2(x) \cdot \ldots \cdot \phi_s(x) = \psi_1(x) \cdot \psi_2(x) \cdot \ldots \cdot \psi_k(x),$$

то $s = k$ и при подходящей нумерации $\phi_i(x) = a_i\psi_i(x)$ для $i = 1, 2, \ldots, s$, где a_i – ненулевые элементы поля $\mathbb{P}$.

Определение 5.3.1. Многочлен, старший коэффициент которого равен 1, называется *унитарным*.

Пусть $p(x)$ — неприводимый делитель многочлена $f(x) \in \mathbb{P}[x]$. Если $p^k(x)$ делит многочлен $f(x)$, $p^{k+1}(x)$ не делит $f(x)$, то $p(x)$ называется *k-кратным неприводимым множителем*. Если $k = 1$, то $p(x)$ называется *простым неприводимым множителем*.

Рассмотрим разложение многочлена $f(x)$ на неприводимые множители:

$$f(x) = \phi_1(x) \cdot \phi_2(x) \cdot \ldots \cdot \phi_s(x).$$

Если вынести за скобки старшие коэффициенты всех неприводимых множителей, а затем собрать совпадающие множители вместе, то получим *каноническое разложение многочлена*

$$f(x) = ap_1^{k_1}(x) \cdot p_2^{k_2}(x) \cdot \ldots \cdot p_n^{k_n}(x),$$

где $p_i(x)$ — унитарные неприводимые попарно взаимно простые многочлены.

С помощью канонических разложений многочленов можно опредлять наибольший общий делитель и наименьшее общее кратное многочленов.

Теорема 5.3.3. 1. Наибольшим общим делителем ненулевых многочленов $f(x)$ и $g(x)$ является произведение всех общих унитарных неприводимых множителей канонических разложений мнногочленов $f(x)$ и $g(x)$, причем каждый множитель берется в наименьшей степени.

2. Наименьшим общим кратным ненулевых многочленов $f(x)$ и $g(x)$ является произведение всех унитарных неприводимых множителей канонических разложений многочленов $f(x)$ и $g(x)$, причем каждый множитель берется в наибольшей степени.

Реализация в системе GAP

Для разложения многочлена на множители в GAP применяется функция Factors. В следующем примере видно, что первый из многочленов является неприводимым над полем рациональных чисел, а второй – нет:

```
gap> f:=1+2*x+3*x^2+5*x^4;
5*x^4+3*x^2+2*x+1
gap> Factors(f);
[ 5*x^4+3*x^2+2*x+1 ]
gap> g:=(x-2)*(x+1/2);
x^2-3/2*x-1
```

```
gap> Factors(g);
[ x-2, x+1/2 ]
```

Однако неприводимость многочленов можно проверить и с помощью функции IsIrreducibleRingElement:

```
gap>IsIrreducibleRingElement(g);
false
gap> IsIrreducibleRingElement(h);
true
```

5.4 Производная многочлена. Корни многочлена

Пусть $\mathbb{P}$ — поле нулевой характеристики, т.е. $na \neq 0 \in \mathbb{P}$ для всех $n \in \mathbb{N}$ и ненулевых элементов a поля $\mathbb{P}$.

Для многочлена

$$f(x) = a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n$$

над полем $\mathbb{P}$ ($deg\ f(x) = n \geq 1$) *производная* определяется следующим образом

$$f'(x) = na_0x^{n-1} + (n-1)a_1x^{n-2} + ... + a_{n-1}.$$

Так как $\mathbb{P}$ — поле нулевой характеристики, то $na_0 \neq 0 \in \mathbb{P}$ и $deg\ f'(x) = n - 1$.

Теорема 5.4.1. Пусть $\mathbb{P}$ — поле нулевой характеристики, $f(x)$ и $g(x)$ многочлены над полем $\mathbb{P}$. Тогда:

1) $a' = 0$ для всех $a \in \mathbb{P}$;
2) $(af(x))' = af'(x)$ для любого $a \in \mathbb{P}$;
3) $(f(x) + g(x))' = f'(x) + g'(x)$;
4) $(f(x) \cdot g(x))' = f'(x) \cdot g(x) + f(x) \cdot g'(x)$;
5) $(f^k(x))' = kf^{k-1}(x) \cdot f'(x)$ для $k \in \mathbb{N}$.

Определение 5.4.1. *Значением многочлена* $f(x) \in \mathbb{P}[x]$ при $x = c \in \mathbb{P}$ называется сумма

$$a_0x^n + a_1c^{n-1} + ... + a_{n-1}c + a_n$$

и обозначается через $f(c)$. Если $f(c) = 0$, то элемент c называют *корнем многочлена* $f(x)$.

Теорема 5.4.2(Безу) Элемент $c \in \mathbb{P}$ является корнем ненулевого многочлена $f(x) \in \mathbb{P}[x]$ тогда и только тогда, когда $(x - c)$ делит $f(x)$.

Определение 5.4.2. Элемент $c \in \mathbb{P}$ называется *k-кратным корнем* многочлена $f(x) \in \mathbb{P}[x]$ или корнем кратности k, если $f(x)$ делится на $(x - c)^k$, но не делится на $(x - c)^{k+1}$. Корень кратности 1 называется *простым*.

Теорема 5.4.3. Элемент $c \in \mathbb{P}$ является k-кратным корнем ненулевого многочлена $f(x) \in \mathbb{P}[x]$ тогда и только тогда, когда

$$f(x) = (x - c)^k q(x) \text{ и } q(c) \neq 0.$$

Теорема 5.4.4. Если $c \in \mathbb{P}$ является k-кратным корнем ненулевого многочлена $f(x)$ над полем нулевой характеристики, то c — $(k-1)$-кратный корень производной $f'(x)$. В частности, c — простой корень многочлена $f(x)$, если $f(c) = 0$ и $f'(c) \neq 0$.

Реализация в системе GAP

Для вычисления производной многочлена в GAP применяется функция Derivative:

```
gap> f:=1+2*x+3*x^2+5*x^4;
5*x^4+3*x^2+2*x+1
gap> Derivative(f);
20*x^3+6*x+2
gap> Derivative(last);
60*x^2+6
gap> Derivative(last);
120*x
gap> Derivative(last);
120
gap> Derivative(last);
0
```

Функция Value(f,a) вычисляет значение многочлена f при заданном

значении переменной $x = a$:

```
gap> g:=(x-2)*(x+1/2);
x^2 - 3/2 * x - 1
gap> Value(g,0);
-1
```

С помощью функции RootsOfUPol можно найти рациональные корни многочленов.

```
f:=1+2*x+3*x^2+5*x^4;
5*x^4+3*x^2+2*x+1
gap> RootsOfUPol(f);    [ ]
gap> g:=(x-2)*(x+1/2);
x^2-3/2*x-1
gap> RootsOfUPol(g);
[ 2, -1/2 ]
```

Теперь проверим второй из результатов:

```
gap> Value(g,2);
0
gap> Value(g,-1/2);
0
```

Замечание. Система GAP позволяет работать с многочленами от нескольких переменных.

5.5 Введение в теорию Галуа

Непустое множество P с двумя бинарными алгебраическими операциями (сложением и умножением) называется *полем*, если выполняются следующие условия:

1) множество P с операцией сложения является абелевой группой;

2) множество $P^* = P \setminus \{0\}$ всех ненулевых элементов с операцией умножения также является абелевой группой;

3) операция сложения связана с операцией умножения законом дистрибутивности: $a(b+c) = ab + ac$ для любых $a, b, c \in P$.

Другими словами, *полем* называется коммутативное кольцо с единицей, в котором каждый ненулевой элемент обратим. Нулевой и единичный элементы поля принято обозначать через 0 и 1 соответственно.

Поскольку множество $P^* = P \setminus \{0\}$ всех ненулевых элементов поля с операцией умножения является абелевой группой, то любое поле P содержит не менее двух элементов, оно всегда содержит элементы 0 и 1, причем в любом поле $0 \neq 1$. Поле не может содержать делителей нуля.

Пример 5.5.1. Примеры полей: $\mathbb{Q}$, $\mathbb{Q}(\sqrt{p}) = \{a + b\sqrt{p} \mid a, b \in \mathbb{Q}\}$, $\mathbb{R}$, причем $\mathbb{Q} \subset \mathbb{Q}(\sqrt{p}) \subset \mathbb{R}$. Кольцо $\mathbb{Z}_p$ классов вычетов, здесь p — простое число.

Характеристика поля

Пусть P — некоторое поле, 0, 1 — нулевой и единичный элементы.

При $n \in \mathbb{N}$ запись $n1$ означает сумму

$$\underbrace{1 + \ldots + 1}_{n}.$$

Возможны две ситуации.

Для любого натурального n всегда $n1 \neq 0$. В этом случае говорят, что поле P имеет *характеристику нуль.*

Для некоторого натурального n выполняется равенство $n1 = 0$. Наименьшее n с этим свойством называют *характеристикой поля* P.

Через $\mathrm{char}\, P$ обозначают характеристику поля P. Ясно, что $\mathrm{char}\, \mathbb{Q} = \mathrm{char}\, \mathbb{Q}(\sqrt{p}) = \mathrm{char}\, \mathbb{R} = 0$, $\mathrm{char}\, \mathbb{Z}_p = p$.

Свойства характеристики поля

1) Если $\mathrm{char}\, P = 0$, то $na \neq 0$ для любых $n \in \mathbb{N}$ и $a \in P \setminus \{0\}$.

2) Если $\mathrm{char}\, P = p \neq 0$, то p — простое число, и $pa = 0$ для всех $a \in P$.

3) Если $\mathrm{char}\, P = p \neq 0$, то $(x + y)^p = x^p + y^p$ для всех $x, y \in P$.

Конечные поля

Поле, в котором конечное число элементов, называется *конечным полем или полем Галуа.* Число элементов конечного поля P называется *порядком поля* и обозначается через $|P|$. Примером конечного поля служит кольцо $\mathbb{Z}_p$ классов вычетов по простому модулю p. В частности, для

любого простого p существует конечное поле из p элементов. Поле Галуа порядка q обозначают через $GF(q)$ или через $\mathbb{F}_q$. Далее вместо $\mathbb{Z}_p$ будем писать $\mathbb{F}_p$, а классы вычетов из $\mathbb{F}_p$ в примерах обозначать без черты.

Свойства полей Галуа

1) Характеристика конечного поля — всегда простое число и если P — конечное поле характеристики p, то $|P| = p^m$ для некоторого $m \in \mathbb{N}$.

2) Любые два конечных поля равных порядков изоморфны между собой.

3) Пусть p и $n \in \mathbb{N}$. Тогда существует поле порядка p^n.

4) Если P — конечное поле, то $P^* = P \setminus \{0\}$ с операцией умножения является циклической группой.

Расширения полей и алгебраические элементы

1. Пусть K — некоторое расширение поля F. Тогда на K можно смотреть как на векторное пространство над F относительно сложения и умножения на элементы из F. Если K конечномерно над F, то K называют конечным расширением поля F. Размерность K над F называют степенью поля K над полем F и обозначают через $[K : F]$.

2. Пусть K — расширение поля F. Элемент $\theta \in K$ называется *алгебраическим над* F, если θ является корнем некоторого ненулевого многочлена из $F[x]$. Если $F = \mathbb{Q}$, то элемент θ называется алгебраическим.

3. *Минимальным многочленом* алгебраического элемента θ над полем F называется ненулевой нормированный многочлен наименьшей степени из $F[x]$, корнем которого является θ. Ясно, что минимальный многочлен единствен и неприводим над F. Его степень называется степенью элемента θ над полем F.

Пример 5.5.2. Докажите, что элемент $1+\sqrt[3]{2}$ является алгебраическим и укажите его минимальный многочлен.

Пусть $\alpha = 1+\sqrt[3]{2}$. Тогда $\alpha-1 = \sqrt[3]{2}$ и $(\alpha-1)^3 = 2$. Раскрывая скобки, получим $\alpha^3 - 3\alpha^2 + 3\alpha - 3 = 0$. Значит, α является корнем многочлена $f(x) = x^3 - 3x^2 + 3x - 3$. По критерию Эйзенштейна многочлен $f(x)$ является неприводимым, а, значит, он минимален для элемента $1+\sqrt[3]{2}$.

Построение полей Галуа

1. Поле $F(\alpha)$ — называется *простым расширением поля* F, если $F(\alpha)$ — наименьшее расширение поля F, содержащее α. Элемент α называется порождающим элементом поля $F(\alpha)$.

2. Простое расширение $F(\alpha)$ называется *простым алгебраическим расширением поля* F, если α является алгебраическим элементом над F.

Выясним из каких элементов состоит простое расширение $F(\alpha)$ поля F и какой оно имеет порядок.

3. Пусть F — поле и $f \in F[x]$. Для того чтобы фактор-кольцо $F[x]/(f)$ было полем необходимо и достаточно, чтобы многочлен f был неприводим над F.

4. Изучим строение кольца $F[x]/(f)$, где f произвольный ненулевой многочлен из $F[x]$. Это кольцо состоит из классов вычетов $[g] = g+(f)$, где $g \in F[x]$. Два класса вычетов $g+(f)$ и $h+(f)$ совпадают в том и только том случае, когда $g \equiv h \ (mod\ f)$, т. е. когда многочлен $g^{-}h$ делится на f. Это равносильно требованию, чтобы g и h давали один и тот же остаток при делении на f. В классе вычетов $g+(f)$ содержится единственный многочлен $r \in F[x]$, для которого $deg\ r < deg\ f$; этот многочлен просто является остатком при делении g на f. Процесс перехода от g к r называется приведением по модулю f. Различные элементы, образующие кольцо $F[x]/(f)$, можно теперь описать явно: а именно это классы вычетов $r+(f)$, где r пробегает все многочлены из $F[]$ степени, меньшей чем $deg\ f$.

Пример 5.5.3. Найдите элементы кольца $\mathbb{F}_2[x]/(x^2+x+1)$ и составьте таблицу их сложения и умножения. Является ли $\mathbb{F}_2[x]/(x^2+x+1)$ полем?

Пусть $f(x) = x^2+x+1$. Многочлены степени 2 и 3 неприводимы над полем F тогда и только тогда, когда он не имеет корней в поле F. Так как $f(0) = 1$, $f(1) = 1$, то $f(x)$ не имеет линейных множителей, значит $f(x)$ неприводим над F_2 и $\mathbb{F}_2[x]/(x^2+x+1)$ — поле. Ясно, что

$$\mathbb{F}_2[x]/(x^2+x+1) = \{a+bx \mid a,\ b \in \mathbb{Z}_2\} = \{0, 1, x, x+1\}.$$

Построим таблицы сложения и умножения для элементов этого поля.

Например, $(x+1)+x = 2x+1 = 0+1 = 1$, $x(x+1) = x^2+x$.

$(+)$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

$(\cdot)$	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

5. Пусть K — расширение поля F и $\theta \in K$ — алгебраический элемент степени n над полем F и $f \in F[x]$ — его минимальный многочлен. Тогда:

a) существует изоморфизм ψ поля $F[x]/(f)$ на поле $F(\theta)$ такой, что $\psi(x) = \theta$ и $\psi(a) = a$ для любого $a \in F$;

b) $1,\ \theta,\ \theta^2,\ \ldots, \theta^{n-1}$ — базис пространства $F(\theta)$ над F и следовательно, $F(\theta)$ — конечное расширение степени n над полем F.

Вывод. Таким образом, если $F(\theta)$ — конечное расширение степени n над полем F порядка q, то $F(\theta) \cong F_{q^n}$.

Пример 5.5.4. Сформируйте поле $\mathbb{F}_4$ при помощи неприводимого над полем $\mathbb{F}_2$ многочлена $f(x) = x^2+x+1$. Постройте таблицы сложения

и умножения для элементов этого поля.

Пусть α — корень многочлена $f(x)$, не принадлежащий полю $\mathbb{F}_2$, т.е. $\alpha^2 = \alpha + 1$. Тогда $1,\ \alpha$ — базис пространства $\mathbb{F}_4$ над полем $\mathbb{F}_2$ и любой элемент из $\mathbb{F}_4$ однозначно представим в виде $a + b\alpha$, где $a,\ b \in \mathbb{F}_2$, т.е. $\mathbb{F}_4 = \{0, 1, \alpha, \alpha + 1\}$. Построим таблицы сложения и умножения для элементов этого поля.

Например, $(\alpha + 1) + \alpha = 2\alpha + 1 = 0 + 1 = 1$, $\alpha(\alpha + 1) = \alpha^2 + \alpha = \alpha + 1 + \alpha = 2\alpha + 1 = 0 + 1 = 1$

$(+)$	0	1	α	$\alpha+1$
0	0	1	α	$\alpha+1$
1	1	0	$\alpha+1$	α
α	α	$\alpha+1$	0	1
$\alpha+1$	$\alpha+1$	α	1	0

$(\cdot)$	0	1	α	$\alpha+1$
0	0	0	0	0
1	0	1	α	$\alpha+1$
α	0	α	$\alpha+1$	1
$\alpha+1$	0	$\alpha+1$	1	α

Примитивные элементы и многочлены

1. Пусть F_q — конечное поле порядка q. Тогда мультипликативная группа F_q^* является циклической.

2. Образующий элемент циклической группы F_q^* называется примитивным элементом поля F_q. Очевидно, что поле F_q содержит $\varphi(q-1)$ примитивных элементов, где φ — функция Эйлера.

3. Нормированный неприводимый многочлен $f \in F_q[x]$ называется примитивным над полем F_q, если f в качестве корня имеет примитивный элемент θ поля F_{q^m}, где $m = degf$. Заметим, что такой f является минимальным многочленом для θ над полем F_q.

Пример 5.5.5. Найдите все примитивные элементы в поле $\mathbb{F}_9$, образованном при помощи неприводимого над полем $\mathbb{F}_3$ многочлена $f(x) = x^2 + 1$.

Ясно, что $\mathbb{F}_9 = \{a + b\alpha \mid a,\ b \in \mathbb{F}_3\} = \{0, 1, 2, \alpha, \alpha+1, \alpha+2, 2\alpha, 2\alpha+1, 2\alpha+2\}$, где α — корень многочлена $f(x)$, не принадлежащий полю $\mathbb{F}_3$. Найдем степени каждого элемента, учитывая, что $\alpha^2 = -1 = 2$. Из таблицы мы видим, что $1+\alpha$, $2+\alpha$, $1+2\alpha$, $2+2\alpha$ являются примитивными элементами в поле $\mathbb{F}_9$.

x	x^2	x^4	x^8		
1	1	1	1	1	
2	1	1	1	2	
α	2	1	1	4	
$\alpha+1$	2α	2	1	8	
$\alpha+2$	α	2	1	8	
2α	2	1	1	4	
$2\alpha+1$	α	2	1	8	
$2\alpha+2$	2α	2	1	8	

Пример 5.5.6. Вычислите $(\alpha+1)^{-3}$ и $(\alpha+1)^{101}$ в поле $\mathbb{F}_4$, образованном при помощи неприводимого над полем $\mathbb{F}_2$ многочлена $f(x) = x^2+x+1$. Здесь, $\alpha \in \mathbb{F}_4$ — корень многочлена $f(x)$.

Поле $\mathbb{F}_4 = \{0, 1, \alpha, \alpha+1\}$. Для вычисления $(\alpha+1)^{-3}$ и $(\alpha+1)^{101}$ воспользуемся таблицей умножения элементов поля из примера 10.3. Ясно, что $(\alpha+1)^{-3} = ((\alpha+1)^{-1})^3 = \alpha^3 = \alpha^2 \cdot \alpha = (\alpha+1)\alpha = 1$. Так как $|\alpha+1| = 3$, то $(\alpha+1)^{101} = (\alpha+1)^{3\cdot 33+2} = (\alpha+1)^{3\cdot 33}\alpha^2 = \alpha^2 = \alpha+1$.

Ответ: 1, $\alpha+1$.

Пример 5.5.7. Решите над полем $\mathbb{F}_8$ с корнем α многочлена $x^3+x+1 \in \mathbb{F}_2[x]$ систему уравнений $\begin{cases} (\alpha^2+1)x + (\alpha^2+\alpha+1)y = 1, \\ \alpha^2 x + (\alpha+1)y = \alpha^2+1. \end{cases}$.

Поскольку $2^3 - 1 = 7$ — число простое, то все ненулевые элементы

поля $\mathbb{F}_8$ есть степени α. Так как характеристика поля $\mathbb{F}_8$ равна 2, то $-1 = 1$ и $\alpha^3 = \alpha + 1$. Тогда $\alpha^4 = \alpha^2 + \alpha$, $\alpha^5 = \alpha^3 + \alpha^2 = \alpha^2 + \alpha + 1$, $\alpha^6 = \alpha^3 + \alpha^2 + \alpha = \alpha^2 + 1$, $\alpha^7 = \alpha^3 + \alpha = \alpha + \alpha + 1 = 1$. Воспользуемся правилом Крамера. Определитель матрицы коэффициентов системы

$$\delta = \begin{vmatrix} \alpha^2 + 1 & \alpha^2 + \alpha + 1 \\ \alpha^2 & \alpha + 1 \end{vmatrix} = \alpha^3 + \alpha^2 + \alpha + 1 + \alpha^4 + \alpha^3 + \alpha^2 = \alpha^2 + 1$$

$$\delta_x = \begin{vmatrix} 1 & \alpha^2 + \alpha + 1 \\ \alpha^2 + 1 & \alpha + 1 \end{vmatrix} = \alpha + 1 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha^2 + \alpha + 1 =$$

$$= \alpha^4 + \alpha^3 = (\alpha^2 + \alpha) + (\alpha + 1) = \alpha^2 + 1;$$

$$\delta_y = \begin{vmatrix} \alpha^2 + 1 & 1 \\ \alpha^2 & \alpha^2 + 1 \end{vmatrix} = \alpha^4 + 1 + \alpha^2 = \alpha + 1.$$

Следовательно,

$$x = \delta_x/\delta = 1; \quad y = \delta_y/\delta = (\alpha + 1)/(\alpha^2 + 1) = 1/\alpha^3 = \alpha^4 = \alpha^2 + \alpha.$$

Ответ: $x = 1$, $y = \alpha^2 + \alpha$.

РАЗДЕЛ 6
ВВЕДЕНИЕ В АЛГЕБРАИЧЕСКУЮ ТЕОРИЮ КОДИРОВАНИЯ

6.1 Введение в криптографию

Предметом изучения *криптографии* (криптография с греческого означает: *крипто* – скрыто, *графо* – пишу) являеся шифрование информации с целью ее защиты от несанкционированного доступа. Из оригинального документа (обычный текст, цифровое изображение, звуковой сигнал и др.), который называют *открытым текстом*, при шифровании образуется его зашифрованная версия, которую называют *шифротекстом*, закрытым текстом, криптограммой. Если шифрование текста – прямая задача, то дешифрование, т.е. превращение шифрованного текста в открытый, – обратная задача.

Основные задачи криптографии:

- обеспечение конфиденциальности – защита информации от ознакомления с ней третьими лицами;
- обеспечение целостности – гарантирование, что информация при хранении или передаче не изменилась;
- аутентификация – это подтверждение подлинности сторон (идентификация);

- обеспечение невозможности отказаться от авторства – предотвращение отказа от факта передачи сообщения.

Стеганография – это наука о скрытой передаче информации путем сохранения в тайне самого факта передачи.

Рассмотрим примеры стенографии.

- На голове раба, которая брилась наголо, записывалось нужное сообщение. Когда волосы раба достаточно отрастали, его отправляли к адресату, который снова брил голову раба и считывал полученное сообщение.
- Запись симпатическими чернилами на предметах одежды, носовых платках и так далее.
- Школьная шпаргалка.
- Скрытая информация в аудио и видео потоках.

Основные виды криптографии:

- *симметрическая криптография* (криптография с закрытым ключом) – ключи для шифрования и дешифрования совпадают или могут легко вычисляться один из другого. Здесь защита информации обеспечена секретностью ключей (нужен тайный канал обмена ключами);

- *ассиметрическая криптография* (криптография с открытым ключом) – если один из ключей (ключ шифрования), называемый открытым ключом, известен всем пользователям, включая криптоаналитика, а другой ключ известен только передающей или принимающей стороне.

В симметрической криптографии выделяют:

- *моноалфавитный шифр* – замена алфавита исходного текста другим алфавитом через замену букв другими буквами или символами;
- *полиалфавитный шифр* – используется несколько алфавитов шифротекста. Каждая буква исходного текста может шифроваться поразному;
- *шифр перестановки.*

Рассмотрим моноалфавитные шифры.

• *Шифр Цезаря.* В I веке н.э. Юлий Цезарь во время войны с галлами, переписываясь со своими друзьями в Риме, заменял в сообщении первую букву латинского алфавита «A» на четвертую «D», вторую «B» – на пятую «E», наконец, последнюю — на третью в соответствии со следующей таблицей:

A	B	C	D	E	F	G	H	I	J	K	L	M
D	E	F	G	H	I	J	K	L	M	N	O	P

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Q	R	S	T	U	V	W	X	Y	Z	A	B	C

В русском варианте эта таблица выглядит так:

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П
Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т

Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В

Пример 6.1.1. Донесение Ю. Цезаря Сенату о победе над понтийским царем «Пришел, увидел, победил» выглядело так: YHQL YLGL YLFL (ЛАТ.). Если бы его зашифровали на русском языке, то получилось бы: ТУЛЫИО ЦЕЛЖЗО ТСДЗЖЛО (РУС.)

• *Пляшущие человечки* (рисунок 6.1). Данный шифр является классическим примером шифра замены, встречающегося в художественной литературе (К. Дойля). Буквы заменены на символические фигурки людей.

«Цель изобретателя этой системы заключалась, очевидно, в том, чтобы скрыть, что эти значки являются письменами, и выдать их за детские рисунки» (Ш.Холмс)

Рис. 6.1: Моноалфавитный шифр «пляшущие человечки»

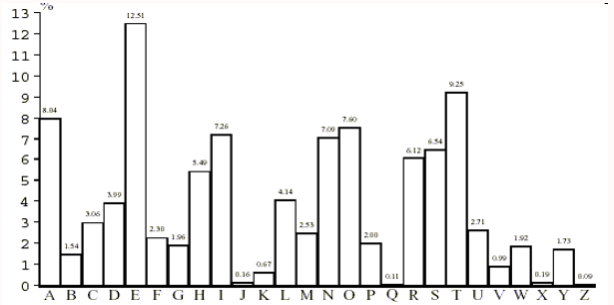


Рис. 6.2: Моноалфавитный шифр «простая замена»

- *Простая замена* (рисунок 6.2).

- *Диск и линейка Энея* (рисунок 6.3). На диске диаметром 10-15 см и толщиной 1-2 см высверливались отверстия по числу букв алфавита. В центре диска – «катушка» с намотанной на ней ниткой. При зашифровании нитка «вытягивалась» с катушки и последовательно протягивалась через отверстия в соответствии с буквами шифруемого текста.

Линейка Энея: в месте прохождения нитки через отверстие завязывался узелок.

Рассмотрим некотрые полиалфавитные шифры.

- *Шифр Виженера* (1586 год, рисунок 6.4).

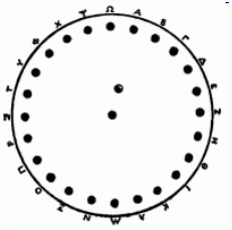


Рис. 6.3: Моноалфавитный шифр «диск и линейка Энея»

В процессе шифрования (и дешифрования) часто используется так называемая «таблица Виженера», которая устроена следующим образом: в первой строке выписывается весь алфавит, в каждой следующей осуществляется циклический сдвиг на одну букву. Так получается квадратная таблица, число сторон которой равно числу столбцов и равно числу букв в алфавите. Чтобы зашифровать какое-нибудь сообщение, поступают следующим образом. Выбирается слово-лозунг и подписывается с повторением над буквами сообщения. Чтобы получить шифрованный текст, находят очередной знак лозунга, начиная с первого в вертикальном алфавите, а ему соответствующий знак сообщения в горизонтальном. На пересечении выделенных столбца и строки находим зашифрованную букву

• *Шифратор Томаса Джефферсона* (1790 год, рисунок 6.5). Деревянный цилиндр разрезается на 36 дисков. Эти диски насаживаются на одну общую ось таким образом, чтобы они могли независимо вращаться на

Рис. 6.4: Шифр Виженера

ней. Для латиницы количество ключей 36!26!, т.е. порядка 10^{60}.

Ключ:

— порядок расположения букв на каждом диске;

— порядок расположения этих дисков на общей оси.

Это изобретение стало предвестником появления так называемых дисковых шифраторов, нашедших широкое распространение в XX веке.

- *Шифровальная машина Энигма* (20-е года XX века, (рисунок 6.6).

Рис. 6.5: Шифратор Томаса Джефферсона

Рис. 6.6: Шифровальная машина Энигма

Перечислим некоторые шифры перестановки.

• *Прибор сцитала* (V в. до н. э., рисунок 6.7)). Является одним из первых физических приборов, реализующих шифр перестановки. Он был изобретен в древней Спарте во времена Ликурга (V в. до н. э.). Для зашифрования текста использовался цилиндр заранее обусловленного диаметра. На цилиндр наматывался тонкий ремень из пергамента, и текст выписывался построчно по образующей цилиндра (вдоль его оси).

Затем ремень сматывался и отправлялся получателю сообщения. Последний наматывал его на цилиндр того же диаметра и читал текст по оси цилиндра. В этом примере ключом шифра является диаметр цилиндра и его длина.

Рис. 6.7: Сцитала

Изобретение дешифровального устройства –«Антисцитала» – приписывается великому Аристотелю. Он предложил использовать конусообразное «копье», на которое наматывался перехваченный ремень; этот ремень передвигался по оси до того положения, пока не появлялся осмысленный текст.

• *Шифрование с использованием перестановок.* Расположим числа от 1 до 5 в двухстрочной записи, в которой вторая строка – произвольная перестановка чисел верхней строки:

1 2 3 4 5
3 2 5 1 4

Эта конструкция носит название подстановки, а число 5 называется ее степенью.

Зашифруем фразу «СВЯЩЕННАЯ РИМСКАЯ ИМПЕРИЯ». В этой фразе 23 буквы. Дополним её двумя произвольными буквами (например, Ь, Э) до ближайшего числа, кратного 5, то есть 25. Выпишем эту дополненную фразу без пропусков, одновременно разбив её на пятизначные группы:

СВЯЩЕ ННАЯР ИМСКА ЯИМПЕ РИЯЬЭ

Буквы каждой группы переставим в соответствии с указанной двухстрочной записью по следующему правилу: первая буква встаёт на третье место, вторая – на второе, третья – на пятое, четвёртая – на первое и пятая – на четвёртое. Полученный текст выписывается без пропусков:

ЩВСЕЯЯННРАКМИАСПИЯЕМЬИРЭЯ

При расшифровании текст разбивается на группы по 5 букв и буквы переставляются в обратном порядке: 1 на 4 место, 2 на 2, 3 на 1, 4 на 5 и 5 на 3. Ключом шифра является выбранное число 5 и порядок расположения чисел в нижнем ряду двухстрочной записи.

- *Магические квадраты* (рисунок 6.8).

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Рис. 6.8: Магический квадрат

Во времена средневековья европейская криптография приобрела сомнительную славу, отголоски которой слышатся и в наши дни. Криптографию стали отождествлять с черной магией, с некоторой формой оккультизма, астрологией, алхимией, каббалой. К шифрованию информации призывались мистические силы. Так, например, рекомендовалось использовать «магические квадраты».

В квадрат размером 4 на 4 (размеры могли быть и другими) вписывались числа от 1 до 16. Его магия состояла в том, что сумма чисел по строкам, столбцам и полным диагоналям равнялась одному и тому же числу – 34. Впервые эти квадраты появились в Китае, где им и была приписана некоторая «магическая сила».

Шифрование по магическому квадрату производилось следующим образом. Например, требуется зашифровать фразу: «ПРИЕЗЖАЮ СЕГОДНЯ». Буквы этой фразы вписываются последовательно в квадрат

согласно записанным в них числам, а в пустые клетки ставятся произвольные буквы.

16У	3И	2Р	13Д
5З	10Е	11Г	8Ю
9С	6Ж	7А	12О
4Е	15Я	14Н	1П

Остановимся более подробно на ассиметрической криптографии (рисунок 6.9).

Математической основой ассиметрической криптографии является нахождение дискретного логарифма (телефонный справочник) и разложение числа на множители.

Пусть дана функция

$$y = f(x) \tag{6.1.1}$$

определенная на конечном множестве X $(x \in X)$, для которой существует обратная функция

$$x = f^{-1}(y) \tag{6.1.2}$$

Функция называется *односторонней*, если вычисление по формуле (6.1.1) – простая задача, требующая немного времени, а вычисление по (6.1.2) – задача сложная, требующая привлечения массы вычислительных ресурсов, уапример, $10^6 - 10^{10}$ лет работы мощного суперкомпью-

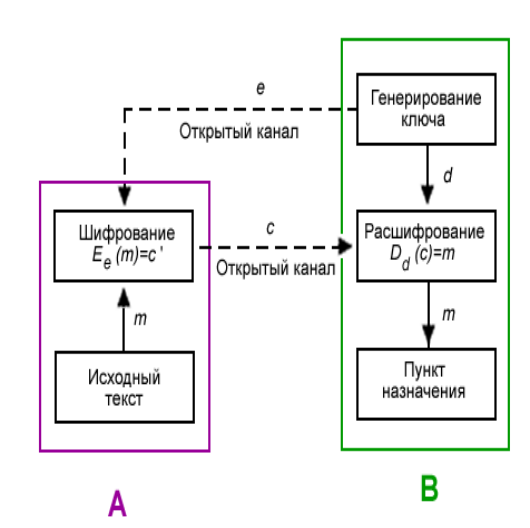


Рис. 6.9: Схематический принцип ассиметрической криптографии

тера. В качестве примера односторонней функции рассмотрим следующую:

$$y = a^x \ mod \ p,$$

где p - некоторое простое число (т.е. такое, которое делится без остатка только на себя и на единицу), а x – целое число из множества $\{1, 2, ..., p-1\}$. Обратная функция обозначается

$$x = \log_a y \ mod \ p$$

и называется *дискретным логарифмом.*

Основные ассиметрические криптосистемы:

- протокол Диффи-Хеллмана;
- шифр Шамира;
- шифр Эль-Гамаля;
- шифр RSA.

Реализация в системе GAP

Пример 6.1.2. Разработайте функцию для шифрования текста используя шифр Цезаря.

```
ShifrCezaria:=function(text)
local i, j, m, shifr, alf1, alf2;
alf1:="abcdefghijklmnopqrstuvwxyz ";
alf2:="defghijklmnopqrstuvwxyz abc";
m:=Size(text);
shifr:=[];
for i in [1..m] do
j:=1;
while text1[i]<>alf1[j] do
j:=j+1;
od;
```

```
Add(shifr,alf2[j]);
od;
return shifr;
end;
```

С помощью разработанной функции зашифруем фразу «i love math».

```
gap> Read("C:/gap4r7/bin/ShifrCezaria.g");
gap> Shifr:=ShifrCezaria("i love math");
"lcoryhcpdwk"
```

Теперь разработаем функцию для дешифрования:

```
ObShifrCezaria:=function(shifr)
local i, j, m, text, alf1, alf2;
alf1:="abcdefghijklmnopqrstuvwxyz ";
alf2:="defghijklmnopqrstuvwxyz abc";
m:=Size(shifr);
text:=[];
for i in [1..m] do
j:=1;
while shifr[i]<>alf2[j] do
j:=j+1;
od;
```

```
Add(text,alf1[j]);
od;
return text;
end;
```

Расшифруем зашифрованный текст:

```
gap> Read("C:/gap4r7/bin/ObShifrCezaria.g");
gap> ObShifrCezaria(Shifr);
"i love math"
```

6.2 Криптосистема Диффи-Хеллмана

Эта криптосистема, приведшая к настоящей революции в криптографии и ее практических применениях, была открыта в середине 70-х годов американскими учеными Диффии Хеллманом. Это первая система, которая позволяла защищать информацию без использования секретных ключей, передаваемых по защищенным каналам.

Рассмотрим сеть связи с N пользователями (N – достаточно большое число). Пусть требуется организовать секретную связь для каждой пары из них. Если мы будем использовать обычную систему распределения секретных ключей, то каждая пара абонентов должна быть снабжена своим секретным ключом, т.е. потребуется $C_N^2 = \frac{N(N-1)}{1} \approx \frac{N^2}{2}$ ключей.

Если абонентов 100, то требуется 5000 ключей, если же абонентов 10^4, то ключей должно быть $5 \cdot 10^7$. Мы видим, что при большом числе абонентов система снабжения их секретными ключами становится очень громоздкой и дорогостоящей.

Диффи и Хеллман решили эту проблему за счет открытого распространения и вычисления ключей.

Пусть строится система связи для абонентов A, B . У каждого абонента есть своя секретная и открытая информация. Для организации этой системы выбирается большое простое число p и некоторое число g, $1 < g < p-1$, такое, что все числа из множества $\{1, 2, \ldots, p-1\}$ могут быть представлены как различные степени $g \bmod p$ (известны различные подходы для нахождения таких чисел g, один из них будет представлен ниже). Числа p и g известны всем абонентам.

Абоненты выбирают большие числа X_A, X_B, которые хранят в секрете (обычно такой выбор рекомендуется проводить случайно, используя датчики случайных чисел). Каждый абонент вычисляет соответствующее число Y, которое открыто передается другим абонентам,

$$\begin{cases} Y_A = g^{X_A} \bmod p, \\ Y_B = g^{X_B} \bmod p, \end{cases}$$

В результате получаем следующую таблицу ключей пользователей в системе Диффи-Хеллмана:

Абонент	Секретный ключ	Открытый ключ
A	X_A	Y_A
B	X_B	Y_B

Допустим, абонент A решил организовать сеанс связи с B, при этом обоим абонентам доступна открытая информация из таблицы 2 . Абонент A сообщает B по открытому каналу, что он хочет передать ему сообщение. Затем абонент A вычисляет величину

$$Z_{AB} = (Y_B)^{X_A} \bmod p$$

(никто другой кроме A этого сделать не может, так как число X_A секретно). В свою очередь, абонент B вычисляет число

$$Z_{BA} = (Y_A)^{X_B} \bmod p$$

Из выше сказанного следует утверждение:

$$Z_{AB} = Z_{BA}.$$

Пример 6.2.1. Пусть $p = 23 = 2 \cdot 11 + 1 (q = 11)$. Выберем параметр g. Попробуем взять g $= 3$. Проверим: $3^{11} \bmod 23 = 1$ и значит, такое g не подходит. Возьмем $g = 5$. Проверим: $5^{11} \bmod 23 = 22$. Итак, мы выбрали параметры $p = 23$, $g = 5$. Теперь каждый абонент выбирает секретное число и вычисляет соответствующее ему открытое число.

Пусть выбраны $X_A = 7$, $X_B = 13$. Вычисляем $Y_A = 5^7 \bmod 23 = 17$, $Y_B = 5^{13} \bmod 23 = 21$. Решили сформировать общий секретный ключ. Для этого A вычисляет $Z_{AB} = 21^7 \bmod 23 = 10$, а B вычисляет $Z_{BA} = 17^{13} \bmod 23 = 10$. Теперь они имеют общий ключ 10, который не передавался по каналу связи.

6.3 Шифр Шамира

Этот шифр, предложенный Шамиром (Adi Shamir), был первым, позволяющим организовать обмен секретными сообщениями по открытой линии связи для лиц, которые не имеют никаких защищенных каналов и секретных ключей и, возможно, никогда не видели друг друга.

Перейдем к описанию системы. Пусть есть два абонента A и B, соединенные линией связи. A хочет передать сообщение m абоненту B так, чтобы никто не узнал его содержание. A выбирает случайное большое простое число p и открыто передает его B.

Затем A выбирает два числа c_A и d_A, такие, что

$$c_A d_A \bmod (p-1) = 1.$$

Эти числа A держит в секрете и передавать не будет.

B тоже выбирает два числа c_B и d_B, такие, что

$$c_B d_B \bmod (p-1) = 1,$$

и держит их в секрете.

Шаг 1. A вычисляет число

$$x_1 = m^{C_A} mod p,$$

где m — исходное сообщение, и пересылает x_1 к B.

Шаг 2. B, получив x_1, вычисляет число

$$x_2 = x_1^{C_B} \bmod p$$

и передает x_2 к A.

Шаг 3. A вычисляет число

$$x_3 = x_2^{d_A} \bmod p$$

и передает его B.

Шаг 4. B, получив x_1, вычисляет число

$$x_4 = x_3^{d_B} \bmod p$$

Свойства протокола Шамира:

1) $x_4 = m$, т.е. в результате реализации протокола от A к B действительно передается исходное сообщение;

2) злоумышленник не может узнать, какое сообщение было передано.

Пример 6.3.1. Пусть A хочет передать B сообщение $m = 10$. A выбирает $p = 23$, $c_A = 7$ $(gcd(7, 22) = 1)$ и вычисляет $d_A = 19$.

Аналогично, B выбирает параметры $c_B = 5$ (взаимно простое с 22) и $d_B = 9$. Переходим к протоколу Шамира.

Шаг 1. $x_1 = 10^7 \bmod 23 = 14$.

Шаг 2. $x_2 = 14^5 \bmod 23 = 15$.

Шаг 3. $x_3 = 15^{19} \bmod 23 = 19$.

Шаг 4. $x_4 = 19^9 \bmod 23 = 10$.

Таким образом, B получил передаваемое сообщение $m = 10$.

Реализация в системе GAP

Пример 6.3.2. Пусть абонент A хочет передать абоненту B сообщение «КОД». Сначала нужно каким-либо способом представить текст сообщения в виде упорядоченного набора чисел. Для простоты предположим, что текст сообщения содержит только заглавными буквами. Первый шаг состоит в замене каждой буквы сообщения числом. Пусть наша таблица замен имеет вид:

А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41

Тогда цифровое представление слова «КОД» имеет вид: 202414.

```
gap> m:=202414;
202414
```

Абонент A выбирает простое число $p = 302417$, $c_A = 2347$

```
gap> p:=302417;
302417
gap> IsPrime(p);
true
gap> cA:=2347;
2347
```

и вычисляет d_A

```
gap> dA:=3000;;
gap> while cA*dA mod (p-1)<>1 do dA:=dA+1 od;
gap> dA;
254483
```

Таким образом, $d_A = 254483$. Абонент A передает абоненту B только число p.

Абонент B выбирает $c_B = 1987$

```
gap> cB:=1987;
1987
```

и вычисляет d_B

```
gap> dB:=2000;;
gap> while cB*dB mod (p-1)<>1 do dB:=dB+1; od;
gap> dB;
280043
```

Таким образом, $d_B = 280043$.

Переходим к протоколу Шамира. Абонент A вычисляет число x_1 и передает его абоненту B.

```
gap> x1:=m^cA mod p;
38370
```

Абонент B, получив число x_1, вычисляет число x_2 и передает его абоненту A.

```
gap> x2:=x1^cB mod p;
99511
```

Абонент A, получив число x_2, вычисляет число x_3 и передает его абоненту B.

```
gap> x3:=x2^dA mod p;
300836
```

Абонент B, получив число x_3, вычисляет число x_4, которое будет

совпадать с передаваемым сообщением m.

```
gap> x4:=x3^ dB mod p;
202414
gap> x4=m;
true
```

6.4 Метод RSA

В 1978 г. Рон Райвест (Ron Rivest), Ади Шамир (Adi Shamir) и Лен Адлеман (Len Adleman) предложили алгоритм с от- крытым ключом. Схема Райвеста–Шамира–Адлемана (RSA) получила широкое распространение.

Опишем процесс шифрования. Исходный текст должен быть переведен в числовую форму, этот метод считается известным. В результате этого текст представляется в виде одного большого числа. Затем полученное число разбивается на части (блоки) так, чтобы каждая из них была числом в промежутке $[0, N-1]$. Процесс шифрования одинаков для каждого блока. Поэтому мы можем считать, что блок исходного текста представлен числом x, $0 \leq x \leq N-1$.

Каждый абонент вырабатывает свою пару ключей. Для этого он генерирует два больших простых числа p и q, вычисляет произведение $N = p \cdot q$. Затем он вырабатывает случайное число e, взаимно простое

со значением функции Эйлера от числа N, $\varphi(N) = (p-1)\cdot(q-1)$ и находит число d из условия $e\cdot d \equiv 1 \bmod \varphi(N)$. Так как НОД$(e, \varphi(N)) = 1$, то такое число d существует и оно единственно. Пару (N, e) он объявляет открытым ключом и помещает в открытый доступ. Пара (N, d) является секретным ключом. Для расшифрования достаточно знать секретный ключ. Числа p, q, $\varphi(N)$ в дальнейшем не нужны, поэтому их можно уничтожить.

Пользователь A, отправляющий сообщение x абоненту B, выбирает из открытого каталога пару (N, e) абонента B и вычисляет шифрованное сообщение $y \equiv x^e \bmod N$. Чтобы получить исходный текст, абонент B вычисляет $y^d \bmod N$. Так как $e \cdot d \equiv 1 \bmod \varphi(N)$, т. е. $e \cdot d = \varphi(N) \cdot k + 1$, где k – целое, то применяя теорему Эйлера, получим: следующее соотношение: $y^d \equiv (x^e)^d \equiv x^{ed} \equiv x^{\varphi(N)\cdot k+1} \equiv (x^{\varphi(N)})^k \cdot x \equiv x \bmod N$.

Пример 6.4.1. Пусть необходимо зашифровать текст «ПОЗНАЙ СЕБЯ». Сначала нужно каким-либо способом представить текст сообщения в виде упорядоченного набора чисел по модулю N. Это еще не процесс шифрования, а только подготовка к нему. Для простоты предположим, что текст сообщения содержит слова, записанные только заглавными буквами. Первый шаг состоит в замене каждой буквы сообщения числом. Пусть наша таблица замен имеет вид:

А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41

Тогда текста «ПОЗНАЙ СЕБЯ» цифровое представление имеет вид: 2524172310199927151141.

Пусть в нашем примере $p = 149$, $q = 157$, тогда $N = 23393$. Поэтому цифро вое представление открытого текста нужно разбить на блоки, меньшие, чем 23393. Одно из таких разбиений выглядит следующим образом:

$$2524 - 1723 - 10199 - 9271 - 511 - 41.$$

Конечно, выбор блоков неоднозначен, но и не совсем произволен. Например, во избежание двусмысленностей, на стадии расшифровки не следует выделять блоки, начинающиеся с нуля.

При расшифровке сообщения получаем последовательность блоков, затем их соединяем вместе и получаем число. После этого числа заменяют буквами в соответствии с таблицей, приведенной выше. Обратим внимание на то, что в этом примере каждую букву кодируем двузначным числом. Это сделано для предотвращения неоднозначности. Если бы мы пронумеровали буквы не по порядку, начиная с 1, т. е. «А» соответствует 1, «Б» соответствует 2 и т. д., то было бы непонятно, что обозначает блок 12: пару букв «АБ» или букву «Л», двенадцатую букву алфавита. Конечно, для кодирования можно использовать любые однозначные соответствия между буквами и числами, например ASCII-кодировку, что

чаще всего это и делается.

Продолжим пример: выбираем $p = 149$, $q = 157$, вычисляем $\varphi(N) = 23088$. Теперь нужно выбрать число e, взаимно простое с $\varphi(N)$. Наименьшее простое, не делящее $\varphi(N)$, равно 5. Положим $e = 5$. Зашифруем первый блок сообщения:

$$\text{вычисляем } 2524^5 \bmod 23393 = 22752;$$
$$\text{далее } 1723^5 \bmod 23393 = 6198;$$
$$10199^5 \bmod 23393 = 14204;$$
$$9271^5 \bmod 23393 = 23191;$$
$$511^5 \bmod 23393 = 10723;$$
$$41^5 \bmod 23393 = 14065.$$

Теперь шифрованный текст имеет вид

$$22752619814204231911072314065.$$

В нашем примере $N = 23393$, $e = 5$. Применив алгоритм Евклида к числам $\varphi(N) = 23088$ и $e = 5$, найдем $d \equiv (e-1) \bmod 23088 \equiv 13853$. Значит для расшифровки блоков шифртекста мы должны возвести этот блок в степень 13583 по модулю 23393. В примере первый блок шифртекста – число 22752, тогда получим $22752^{13853} \bmod 23393 \equiv 2524$.

Разбиение числа на блоки можно произвести различными способами. При этом промежуточные результаты зависят от способа разбиения, однако конечный результат – не зависит.

Лабораторная работа №1. Основы работы с ситемой GAP

Изучите теоретический материал пунктов «Краткая характеристика, история и обзор возможностей системы GAP», «Начало работы в GAP» и «Язык программирования GAP».

Особенности работы в GAP:

- Одна команда может занимать несколько строк, последняя из которых заканчивается точкой с запятой.
- В GAP имеет значение регистр текста.
- При некоторых ошибках на экран выводится промежуточное приглашение brk>. Для выхода из него нужно ввести команду quit;.
- В GAP есть возможность работать с историей команд. Если набрать в командной строке какой-либо символ (последовательность символов), а затем нажимать клавиши управлепния курсором, то Вы будете видеть только те из ранее введенных команд, которые начинались с введенного символа (последовательности символов).
- Перемещеться по содержимому командной строки можно с помощью клавиш перемещения курсора влево и вправо. Можно удалять

символы с помощью клавиш Delete и Backspace. Для быстрого перемещения в конец и начало строки можно использовать клавиши Home и End.

- В GAP имется возможность копировать и вставлять текст. Вставить в командную строку текст из буфера обмена можно используя стандартные способы в ставки в окне MS-DOS и используя сочетание клавиш Shift-Ins в окне RXVT.

Задание 1. Найдите каталог gap4r4, в котором инсталлирована система GAP на локальном или сетевом диске (например, с помощью FAR или Проводника). Найдите в каталоге gap4r4/bin командные файлы gap.bat и gaprxvt.bat. Теперь запустите систему GAP либо с помощью файла gap.bat для работы в окне командной строки Windows (окне MS-DOS), либо с помощью файла gaprxvt.bat для работы в окне оболочки RXVT.

Простейшие вычисления можно выполнять, запуская систему так, как указано в выше. Однако, в этом случае при чтении и записи файлов нужно будет указывать полный путь к ним. Эффективнее будет создать рабочий каталог в том разделе диска, где Вы имеете соответствующие права доступа, и скопировать туда файлы gap.bat и gaprxvt.bat. Выполните эти инструкции, создав свой рабочий каталог (который можно назвать, например, gap).

Задание 2. Выполните простейшие вычисления, введя следующие команды:

```
352/182;
2*(15+256)/17;
2 ^ 64;
2 ^ mod 100;
3 in [1,2,3] ;
2*2 >= 4;
```

Одна команда может занимать несколько строк, последняя из которых заканчивается точкой с запятой. Таким образом, если Вы забыли поставить точку с запятой в конце строки и уже нажали клавишу Enter, Вы можете поставить точку с запятой в следующей строке, а затем нажать Enter еще раз. Попробуйте ввести следующую многострочную команду:

```
155/4545+
1234*5678+
Factorial(100)+
Sum([1..100]);
```

Задание 3. Попробуйте выделить в окне браузера (т.е. MS Internet Explorer, Netscape и т.п.) и скопировать в буфер обмена команды, приведенные выше, а затем перейти в окно GAP и вставить их в командную строку (в окне MS-DOS используйте стандартные способы вставки, а в

окне RXVT используйте сочетание клавиш Shift-Ins). Затем попробуйте выделить и скопировать текст из окна GAP (в окне MS-DOS используйте стандартные средства, в окне RXVT выделяйте текст мышью, а для копирования используйте Ctrl-Ins) и вставить его в текстовый файл (редактируемый, например, с помощью FAR или Блокнота).

Задание 4. Одной из составных частей системы GAP является ее документация. С помощью Проводника откройте каталог gap4r4/doc. В нем Вы обнаружите подкаталог htm, в котором нужно открыть файл index. htm - это стартовый файл для просмотра документации в HTML-формате.

Для быстрого обращения к документации создайте в своем рабочем каталоге ярлык, указывающий на файл index.htm, после чего откройте его с помощью данного ярлыка и ознакомьтесь с названиями пяти основных разделов документации. Перейдите в раздел «Индекс» и найдите с его помощью описание функций Factorial и Sum. Вы можете скопировать приведенные в документации примеры и выполнять их в GAP так, как это было описано в предыдущем задании.

При полной инсталляции системы каталог gap4r4/doc также содержит документацию и в других форматах. В частности, он содержит другие подкаталоги, наименования которых соответствуют пяти основным разделам документации, в которых можно найти эти разделы в формате PDF, более удобном при печати документации (учтите, что центральный

раздел документации - Reference Manual - занимает в формате PDF почти тысячу страниц!).

Альтернативным вариантом использования документации является подстрочная справка, которую можно вызвать прямо из командной строки GAP. Это удобно, если в дальнейшем не предвидится активное перемещение по гиперссылкам в документации, а также может быть полезно при удаленном подключении или в случае, когда ресурсы компьютера ограничены. Наберите в командной строке ?Factorial (без точки с запятой) для отображения справки по данной функции.

Задание 5. Историю работы с системой можно сохранить в текстовом файле (т.наз. файле протокола). Введите команду

```
LogTo("logfile.txt");
```

После этого все введенные Вами команды и результаты их работы, отображаемые на экране, будут дублироваться в файле с именем logfile.txt, который содержится в Вашем рабочем каталоге.

Теперь задайте переменную n, в которой сохраните номер своего варианта, например:

```
n:=20;
```

Затем последовательно введите следующие команды:

```
a:=2 ^ (n+1)-1;
IsPrime(a);
Factors(a);
x:=n+10;
Factors(Factorial(x));
Phi(x);
Sigma(x);
Tau(x);
```

Теперь закройте файл протокола с помощью команды

```
LogTo();
```

и просмотрите его с помощью, например, FAR или Проводника.

Задание 6. Вычислите число возможных комбинаций кубика-рубика $3\times3\times3$, зная что оно вычисляется по формуле $(8!\times3^{8-1})\times(12!\times2^{12-1})/2$. Однако эта формула не учитывает то, что ориентация центральных квадратов может быть разной. С учётом ориентации центральных квадратов количество состояний возрастает в $\frac{4^6}{2}$ раз.

Лабораторная работа №2. Списки. Целые числа. НОД целых чисел. Арифметические функции

Данная лабораторная работа предназначена для изучения приемов работы со списками на примере действий над целыми числами. Подробные сведения по данным темам содержатся: «Списки», «Простые числа. Разложение натуральных чисел на простые множители. Числовые функции»

В зависимости от конкретной задачи, при выполнении работы полезными могут оказаться следующие функции и операции (детальное их описание см. в документации):

Collected(list) – возвращает новый список newlist, который для каждого элемента x исходного списка list содержит соответствующий ему список из двух элементов, первый из которых является самим элементом , а второй показывает кратность его вхождения в список list.

Combinations(list[,k]) – возвращает множество всевозможных комбинаций (неупорядоченных наборов без повторений), составленных изk элементов списка list (который может даже содержать одинаковые элементы несколько раз). Если k не указано, возвращаются все возможные комбинации, составленные из элементов списка list.

DivisorsInt(n) – возвращает список натуральных делителей целого числа n.

FactorsInt(n) – возвращает разложение целого числа n на простые

множители в виде их списка.

PrimePowersInt(n) – возвращает разложение целого числа n на простые множители, с указанием степеней входящих в это разложение простых чисел.

Filtered(list, x->f(x)) – возвращает список тех элементов из списка list, для которых выполняется условие $f() = true$.

ForAll(list,x->f(x)) – проверяет, что для каждого элемента x из списка list выполняется условие $f(x) = true$.

ForAny(list, x->f(x)) – проверяет, что существует хотя бы один элемент x из списка list, для которого выполняется условие $f(x) = true$.

Gcd(list) или *Gcd*(a_1,a_2,...,a_N) – вычисляет наибольший общий делитель целых чисел a_1, a_2,... или целых чисел из списка list.

Length(list) – определяет длину списка list.

a *mod* b – возвращает остаток от деления a на b.

Phi(n) – вычисляет функцию Эйлера $\varphi(n)$, т.е. количество чисел ряда $0, 1, ..., a-1$ взаимно простых с a.

Sigma(n) – вычисляет функцию $\sigma(n)$, т.е. сумму натуральных делителей числа n..

Tau(n) – вычисляет функцию $\tau(n)$, т.е. число натуральных делителей числа n.

Product(list) – вычисляет произведение всех элементов списка list.

Sum(list) – вычисляет сумму всех элементов списка list.

Задание 1. Разложите на простые множители число $n!$.

Вариант 1. $n = 20$. *Вариант 2.* $n = 30$.

Вариант 3. $n = 45$. *Вариант 4.* $n = 60$.

Вариант 5. $n = 70$. *Вариант 6.* $n = 82$.

Задание 2. Найдите показатель степени числа p в каноническом разложении числа 1000!.

Вариант 1. $p = 3$. *Вариант 2.* $p = 5$.

Вариант 3. $p = 7$. *Вариант 4.* $p = 11$.

Вариант 5. $p = 13$. *Вариант 6.* $p = 17$.

Задание 3. Найдите количество целых положительных чисел, не превосходящих n и не делящихся ни на одно из простых чисел a, b, c.

	n	a	b	c
Вариант 1.	2000	5	7	13
Вариант 2.	2150	3	11	17
Вариант 3.	4152	11	7	5
Вариант 4.	6122	2	3	4
Вариант 5.	1800	3	7	11
Вариант 6.	1245	3	11	23

Задание 4. *Вариант 1.* Найдите количество целых положительных чисел, не превосходящих 100 и взаимно простых с 36.

Вариант 2. Найдите количество целых положительных чисел, не превосходящих 12317 и взаимно простых с 1575.

Вариант 3. Найдите количество натуральных чисел, меньших числа 300 и имеющих с ним наибольшим общим делителем число 20.

Вариант 4. Найдите количество натуральных чисел, меньших числа 1665 и имеющих с ним наибольшим общим делителем число 37.

Вариант 5. Найдите количество натуральных чисел, меньших числа 1476 и имеющих с ним наибольшим общим делителем число 41.

Вариант 6. Найдите количество целых положительных чисел, не превосходящих 1000 и не взаимно простых с 363.

Задание 5 *Вариант 1.* Найдите целое положительное число, зная, что оно имеет только два простых делителя, число всех делителей равно 6, а сумма всех делителей равна 28.

Вариант 2. Найдите целое положительное число, произведение всех делителей которого равно 5832.

Вариант 3. Покажите, что число 496 является совершенным, т.е. равным сумме своих собственных делителей.

Вариант 4. Найдите целое положительное число, зная, что оно имеет только два простых делителя, число всех делителей равно 6, а сумма всех делителей равна 28.

Вариант 5. Найдите целое положительное число, произведение всех делителей которого равно 5832.

Вариант 6. Покажите, что число 496 является совершенным, т.е. равным сумме своих собственных делителей.

Задание 6. Найдите все делители числа n.

Вариант 1. $n = 360$. *Вариант 2.* $n = 375$.
Вариант 3. $n = 957$. *Вариант 4.* $n = 988$.
Вариант 5. $n = 960$. *Вариант 6.* $n = 532$.

Задание 7. Определите сколькими нулями заканчивается десятичная запись числа $\varphi(a!)$?

Вариант 1. $a = 92$. *Вариант 2.* $a = 72$.
Вариант 3. $a = 88$. *Вариант 4.* $a = 104$.
Вариант 5. $a = 64$. *Вариант 6.* $a = 90$.

Задание 8. Решите уравнение на интервале $[0; 120]$.

Вариант 1. $\varphi(x) = 8$. *Вариант 2.* $\varphi(x) = 12$.
Вариант 3. $\varphi(x) = 24$. *Вариант 4.* $\varphi(x) = 16$.
Вариант 5. $\varphi(x) = 18$. *Вариант 6.* $\varphi(x) = 36$.

Задание 9. Найдите $n < 50000$, если известен его делитель m и значение $\tau(n)$.

Вариант 1. $m = 135$, $\tau(n) = 21$. *Вариант 2.* $m = 104$, $\tau(n) = 15$.
Вариант 3. $m = 88$, $\tau(n) = 21$. *Вариант 4.* $m = 75$, $\tau(n) = 14$.
Вариант 5. $m = 99$, $\tau(n) = 10$. *Вариант 6.* $m = 40$, $\tau(n) = 33$.

Задание 10. Пусть $n < 1000$. Найдите $\tau(n^3)$, если известно значение $\tau(n^2)$.

Вариант 1. $\tau(n^2) = 77$. *Вариант 2.* $\tau(n^2) = 75$.
Вариант 3. $\tau(n^2) = 85$. *Вариант 4.* $\tau(n^2) = 91$.
Вариант 5. $\tau(n^2) = 93$. *Вариант 6.* $\tau(n^2) = 95$.

Лабораторная работа №3. Функции. Условный оператор IF, циклы FOR и WHILE

Задание 1. Изучите теоретический материал по темам «Матрицы и операции над ними» и «Определитель матрицы». Для выполнения задания обязательно примените цикл FOR.

Вариант 1. Разработайте функцию для вычисления определителя матрицы A второго порядка, входным параметром которой является матрица A, а выходным параметром – определитель вычисленный по формуле $det\ A = a_{11}a_{22} - a_{12}a_{21}$.

*Вариант 2.*Разработайте функцию для вычисления определителя матрицы A третьего порядка, входным параметром которой является матрица A, а выходным параметром – определитель вычисленный по правилу треугольника.

Вариант 3. Разработайте функцию для вычисления произведения двух матриц второго порядка, входными параметрами которой являются матрицы A и B, а выходным параметром – их произведение $G = AB$.

Вариант 4. Разработайте функцию для транспонирования матрицы третьего порядка, входным параметром которой является матрица A, а

выходным параметром – матрица A^T.

Вариант 5. Разработайте функцию для вычисления скалярного произведения двух векторов, заданных своими координатами, входными параметрами которой являются вектора и b, а выходным параметром – их скалярное произведение ab.

Вариант 6. Разработайте функцию для вычисления векторного произведения двух векторов, заданных своими координатами, входными параметрами которой являются вектора и b, а выходным параметром – их векторное произведение $[a, b]$.

Задание 2. Изучите теоретический материал по темам «Кольцо многочленов», «Деление в кольце многочленов» и «Производная многочлена. Корни многочлена». Для выполнения задания обязательно примените условный оператор IF.

Вариант 1. Разработайте функцию, которая определяет являются ли два заданных многочлена f и g взаимно простыми и, в случае положительного ответа возвращает их произведение, а в случае отрицательного ответа – их произведение деленное на их НОД.

Вариант 2. Разработайте функцию, которая определяет, все ли коэффициенты многочлена f являются четными. В случае положительного ответа возвращает многочлен $\frac{1}{2}f$, а в случае отрицательного ответа – исходный многочлен.

Вариант 3. Разработайте функцию, которая определяет разлагается

ли данный многочлен в произведение линейных множителей. В случае положительного ответа возвращает их произведение, а в случае отрицательного ответа – единицу.

Вариант 4. Разработайте функцию, которая определяет равен ли нулю свободный член многочлена f. В случае положительного ответа возвращает многочлен f/x, а в случае отрицательного ответа – исходный многочлен.

Вариант 5. Разработайте функцию, которая определяет совпадают ли степени многочленов. В случае положительного ответа возвращает их сумму, а в случае отрицательного ответа – их произведение.

Вариант 6. Разработайте функцию, которая определяет равна ли нулю производная многочлена в точке a. В случае положительного ответа возвращает значение многочлена в точке a, а в случае отрицательного ответа – значение производной в точке a.

Задание 3. Для выполнения задания обязательно примените цикл WHILE.

Вариант 1. Разработайте функцию, которая для заданной подстановки s определяет минимальное натуральное число k, такое что sk коммутирует с заданной подстановкой t.

Вариант 2. Разработайте функцию, которая для заданной подстановки s определяет минимальное натуральное число k, такое что sk переводит заданное натуральное число n в заданное натуральное число m,

и возвращает fail, если такого числа k не существует.

Вариант 3. Разработайте функцию, которая для заданной подстановки s определяет минимальное натуральное число k, которое она оставляет на месте.

Вариант 4. Разработайте функцию, которая для заданной подстановки s определяет минимальное натуральное число k, такое что sk оставляет на месте заданное натуральное число n.

Вариант 5. Разработайте функцию, которая для заданной подстановки s определяет минимальное натуральное число k, такое что количество натуральных чисел, перемещаемых подстановкой sk, не превосходит заданного натурального числа n.

Вариант 6. Разработайте функцию, которая для заданной подстановки s определяет минимальное натуральное число k, такое что sk оставляет на месте единицу.

Задание 4. Перед выполнением задания изучите теоретический материал по теме «Отношение сравнения в кольце $\mathbb{Z}$».

Вариант 1. Составьте функцию, которая для произвольных a и b на интервале [1..100] будет находить все решения уравнения $a\varphi(x) = bx$.

Вариант 2. Составьте функцию, которая для каждого m будет вычислять идемпотенты кольца Z_m.

Вариант 3. Составьте функцию, которая для произвольного многочлена $ax^2 + bx +$ вычисляет все его корни в кольце Z_m.

Вариант 4. Составьте функцию, которая для произвольных a, b и m находит решения сравнения $ax = b \bmod m$.

Вариант 5. Составьте функцию, которая находит решение произвольной системы сравнений

$$\begin{cases} a_1 x = b_1 \bmod m_1; \\ a_2 x = b_2 \bmod m_2; \\ a_3 x = b_3 \bmod m_3. \end{cases}$$

Вариант 6. Составьте функцию, которая для произвольного многочлена $f(x) = ax^2 + bx +$ находит многочлен, корнями которого являются α^{-1} и β^{-1}, где α и β – корни многочлена $f(x)$ кольце Z_m.

Лабораторная работа №4. Структура и свойства группы

Перед выполнением лаболаторной работы изучите теоретический материал раздела «Элементы теории групп в системе GAP».

Задание 1. Данное задание предназначено для изучения некоторых приемов работы с элементами групп.

Вариант 1. Разработайте функцию, которая возвращает множество порядков элементов заданной группы. Указание: используте функции AsList, Order, Set.

Вариант 2. Разработайте функцию, которая возвращает множество элементов заданной группы, имеющих порядок, равный заданному числу k. Указание: используйте функции AsList, Order.

Вариант 3. Разработайте функцию, которая возвращает множество порядков классов сопряженных элементов заданной группы. Указание: используйте функцию ConjugacyClasses.

Вариант 4. Разработайте функцию, которая возвращает множество порядков классов сопряженных элементов заданной группы. Указание: используйте функцию ConjugacyClasses.

Вариант 5. Разработайте функцию для вычисления количества элементов каждого порядка в заданной группе.

Вариант 6. Проверьте, выполняется ли в группе подстановок S_3 тождество $x^6 = 1$.

Задание 2. Данное задание предназначено для изучения работы с подгруппами.

Вариант 1. Разработайте функцию, которая для заданной группы возвращает множество порядков всех ее подгрупп, полученное как множество порядков представителей ее классов сопряженных подгрупп. Указание: используйте функции ConjugacyClassesSubgroups, Representative.

Вариант 2. Разработайте функцию, которая для заданной группы возвращает список простых делителей порядка группы с указанием порядка и количества соответствующих Силовских p-подгрупп. Указание: используйтеь функции SylowSubgroup, ConjugacyClassSubgroups.

Вариант 3. Разработайте функцию, которая для заданной группы возвращает множество порядков ее максимальных подгрупп. Указание:

используйте функции Size, MaximalSubgroups.

Вариант 4. Разработайте функцию, которая для заданной группы возвращает множество порядков ее нормальных подгрупп. Указание: используйте функции Size, NormalSubgroups.

Вариант 5. Составьте функцию, которая для заданной группы вычисляет подгруппу Фраттини, т.е. пересечение всех ее максимальных подгрупп. Указание: используйте функции Intersection, MaximalSubgroups.

Вариант 6. Разработайте функцию, которая для заданной группы определяет порядок ее фактор-группы по коммутанту. Указание: используйте функции Size и DerivedSubgroup.

Задание 3. Данное задание предназначено для изучения работы с библиотекой конечных групп системы GAP. Необходимо сначала выбрать группы из библиотеки по указанному критерию (порядок группы в сочетании с некоторым свойством), а затем для каждой из этих групп определить указанное свойство, и вывести результаты на экран в виде таблицы с указанием номера соответствующей группы в библиотеке конечных групп системы GAP.

Вариант 1. Выберите все нециклические 2-группы порядка 32. Определите их количество. Каждую из них идентифицируйте с помощью функции GroupId и вычислите ее класс нильпотентности с помощью функции LowerCentralSeries. Результат необходимо вывести в виде таблицы.

Вариант 2. Выберите все 2-группы порядка 32, порядок коммутанта которых равен 8. Определите их количество. Идентифицируйте каждую с помощью функции GroupId и вычислите длину ее ряда коммутантов с помощью функции DerivedSeries. Результат необходимо вывести в виде таблицы.

Вариант 3. Выберите все нециклические 3-группы порядка 27. Определите их количество. Каждую из них идентифицируйте с помощью функции GroupId и вычислите ее класс нильпотентности с помощью функции LowerCentralSeries. Результат необходимо вывести в виде таблицы.

Вариант 4. Выберите все неабелевы 3-группы порядка 27. Определите их количество. Идентифицируйте каждую из них с помощью функции GroupId и вычислите порядок ее центра. Результат необходимо вывести в виде таблицы.

Вариант 5. Выберите все 3-группы порядка 81 с коммутантом порядка 9. Определите их количество. Идентифицируйте каждую из них с помощью функции GroupId и вычислите длину ее ряда коммутантов с помощью функции DerivedSeries. Результат необходимо вывести в виде таблицы.

Вариант 6. Выберите все неабелевы 2-группы порядка 64. Определите их количество. Идентифицируйте каждую из них с помощью функции GroupId и вычислите порядок ее центра. Результат необходимо вывести в виде таблицы.

Задание 4. *Вариант 1.* Разработайте функцию для вычисления производной длины разрешимой группы.

Вариант 2. Разработайте функцию для вычисления нильпотентной длины разрешимой группы.

Вариант 3. Разработайте функцию для вычисления p-ранга разрешимой группы.

Вариант 4. Разработайте функцию, которая определяет является ли группа дисперсивной по Оре.

Вариант 5. Разработайте функцию, которая определяет является ли группа бициклической, т.е. произведением двух своих циклических подгрупп.

Вариант 6. Разработайте функцию, которая определяет является ли произведение двух групп группой.

Лабораторная работа №5. Шифрование методом RSA

Изучите теоретический материал по темам «Элементы теории чисел в системе GAP» и «Метод RSA».

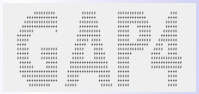
Задание 1. Зашифруйте свою фамилию и имя методом RSA, используя указанные пары простых чисел.

Вариант 1. p = 8297, q = 8293;

Вариант 2. p = 9281, q = 9283;

Вариант 3. p = 9341, q = 9343;
Вариант 4. p = 8291, q = 8293;
Вариант 5. p = 7949, q = 7951;
Вариант 6. p = 7211, q = 7213;
Вариант 7. p = 6701, q = 6703;
Вариант 8. p = 9929, q = 9931;
Вариант 9. p = 6359, q = 6361;
Вариант 10. p = 6299, q = 6301.

Задание 2. Произведите дешифрование.



Тесты для самоконтроля

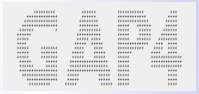
ТЕСТ №1.
ТЕСТ №2.
ТЕСТ №3.
ТЕСТ №4.
ТЕСТ №5.
ТЕСТ №6.

Вопросы к зачету

1. Группы. Способы задания групп. Подгруппы.
2. Циклические группы, подгруппы циклической группы.
3. Нормализатор и централизатор группы.
4. Прямое пргоизведение групп.
5. Полупрямое произведение групп.
6. Классы групп.
7. Группы малых порядков.
8. Инварианты разрешимых групп.
9. Делимость целых чисел.
10. Наибольший общий делитель (НОД). Алгоритм Евклида.
11. Наменьшее общее кратное (НОК)
12. Простые числа. Разложение натуральных чисел на простые множители.
13. Числовые функции.
14. Функция Эйлера и ее приложения.
15. Сравнения с одним неизвестным.
16. Системы сравнений. Китайская теорема об остатках.
17. Матрицы и операции над ними.
18. Определитель матрицы.
19. Системы линейных уравнений. Метод Гаусса.
20. Системы линейных уравнений. Правило Крамера.

21. Системы линейных уравнений. Матричный метод.
22. Кольцо многочленов.
23. Деление в кольце многочленов.
24. Разложение многочленов на неприводимые множители.
25. Производная многочлена.
26. Корни многочлена.
27. Основные понятия криптографии.
28. Метод RSA.

Демонстрационный варинт нулевого билета к зачету



БИЛЕТ №0

Дисциплина «Компьютерная алгебра»
Специальность «Математика и информатика»

1. Группы. Способы задания групп. Подгруппы.

2. Разработайте функцию, которая определяет, равна ли нулю сумма коэффициентов заданного многочлена f, и в случае положительного ответа возвращает исходный многочлен, а в случае отрицательного – многочлен, полученный из многочлена f вычитанием из него суммы своих коэффициентов.

Список использованной и рекомендованной литературы

1. Gallian, J.A. Abstract algebra with GAP / J.A. Gallian. — Brooks / Cole, 2010. — 98 p.
2. Hulpke, A. Abstract algebra in GAP / A. Hulpke. — Fort Collins, 2010. — 136 p.
3. Joyner, D. Applied abstract algebra / D. Joyner, R. Kreminski, J. Turisco. — The Johns Hopkins University Press, 2004. — 344 p.
4. The GAP Group, GAP — Groups, Algorithms, and Programming, Version 4.4; 2016. — Mode of access: http://www.gap-system.org.
5. Глухов, М.М. Алгебра. Том 1-2 / М.М. Глухов, В.П. Елизаров, А.А. Нечаев — М.: Гелиос АРВ, 2003. — 416 с.
6. Грицук, Д.В. Компьютерная алгебра. Курс лекций / Д.В. Грицук, А.А. Трофимук; Брест. гос. ун-т им. А.С. Пушкина. — Брест: БрГУ, 2017. — 112 с.
7. Кабанов В.В. Конечные поля (электронное издание) – Екатеринбург: УрГУ, 2008
8. Коновалов, А. Б. Система компьютерной алгебры GAP 4.7 (Brief GAP Guidebook in Russian) / А. Б. Коновалов. — Режим доступа: http://www.gap-system.org/ukrgap/gapbook/manual.pdf

9. Кострикин, А.И. Введение в алгебру / А.И. Кострикин. — М. : Наука, 1977. — 495 с.

10. Курош, А.Г. Курс высшей алгебры / А.Г. Курош. — М. : Наука, 1971. — 424 с.

11. Лидл, Р. Конечные поля. В 2-х томах. Том 1. / Р. Лидл, Г. Нидеррайтер. — М. : Мир, 1988. — 430 с.

12. Милованов, М.В. Алгебра и аналитическая геометрия. Часть 1 / М.В. Милованов, Р.И. Тышкевич, А.С. Феденко. — Мн.: Амалфея, 2001. — 400 с.

13. Монахов, В. С. Введение в теорию конечных групп и их классов / В. С. Монахов. — Минск : Выш. шк., 2006. — 207 с.

14. Монахов, В. С. Алгебра и теория чисел : учеб. пособие : в 2 ч. / В. С. Монахов, А. В. Бузланов. – Минск : Изд. центр БГУ, 2007.

15. Монахов, В. С. Числовые функции и классы вычетов : практикум / В. С. Монахов, А. А. Трофимук ; Брест. гос. ун-т им. А.С. Пушкина. — Брест : БрГУ, 2012. — 88 с.

16. Рябко, Б.Я. Криптографические методы защиты информации: Учебное пособие для вузов / Б.Я. Рябко, А.Н. Фионов — М.: горячая линия — Телеком. — 229 с.

17. Сенашов, В. И. Основы теории групп. Версия 1.0 [Электронный ресурс] : курс лекций / В. И. Сенашов, А. В. Тимофеенко, В. П. Шунков. — Красноярск : ИПК СФУ, 2008. — Электрон. дан. (1 Мб).