Министерство образования Республики Беларусь
Учреждение образования «Витебский государственный
университет имени П.М. Машерова»
Кафедра информационных технологий и управления бизнесом

КОМПЬЮТЕРНАЯ АЛГЕБРА

Методические рекомендации

Витебск
ВГУ имени П.М. Машерова
2021

УДК [512.6+519.85](075.8)
ББК 22.195.3я73
К63

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 1 от 27.10.2021.

Составитель: доцент кафедры информационных технологий и управления бизнесом ВГУ имени П.М. Машерова, кандидат физико-математических наук **Е.А. Витько**

Р е ц е н з е н т ы :
доцент кафедры алгебры и методики преподавания математики
ВГУ имени П.М. Машерова,
кандидат физико-математических наук *А.П. Мехович;*
старший преподаватель кафедры математики
и информационных технологий УО «ВГТУ» *А.В. Коваленко*

К63 **Компьютерная алгебра :** методические рекомендации / сост. Е.А. Витько. – Витебск : ВГУ имени П.М. Машерова, 2021. – 34 с.

Данное издание подготовлено в соответствии с учебной программой дисциплины «Компьютерная алгебра» II ступени высшего образования. Приведены варианты заданий лабораторных работ по дисциплине и основные теоретические сведения, необходимые для их решения.

УДК [512.6+519.85](075.8)
ББК 22.195.3я73

Содержание

Введение 4
История системы GAP. Начало работы в GAP 5
Лабораторная работа 1 «Основы работы с системой GAP» 8
Язык программирования GAP 10
Лабораторная работа 2 «Списки» 16
Лабораторная работа 3 «Функции» 19
Условный оператор и операторы циклов в GAP 21
Лабораторная работа 4 «Ветвящиеся и циклические программы» 24
Элементы теории групп в GAP 26
Лабораторная работа 5 «Группы, свойства элементов группы, подгруппы» 30
Литература 33

Введение

Данное издание содержит базовые сведения о системе компьютерной алгебры GAP (от англ. Groups, Algorithms, Programming – Группы, Алгоритмы, Программирование). GAP – система, задуманная как инструмент вычислительной теории групп, и впоследствии распространившаяся на смежные разделы алгебры. Система включает библиотеку функций (более 4000), в которой реализованы разнообразные алгоритмы. Пользователю доступны различные комбинаторные функции, элементарные теоретико-числовые функции, функции для работы с множествами и списками. Кроме того, GAP предоставляет пользователю библиотеку данных, которая содержит, например, библиотеку всех групп порядка не более 2000 (за исключением групп порядка 1024). Группы могут быть заданы в различной форме, например, как группы подстановок, матричные группы, группы, заданные порождающими элементами и определяющими соотношениями.

Методические рекомендации подготовлены в соответствии с учебной программой дисциплины «Компьютерная алгебра» специальности «Математика и компьютерные науки» II ступени высшего образования.

При изложении теоретических вопросов теории групп в издании используется материал учебного пособия В.С. Монахова [3], при описании методов работы с системой GAP – электронного учебно-методического комплекса [1] и пособия [2].

История системы GAP. Начало работы в GAP

Разработка системы компьютерной алгебры GAP (Groups, Algorithms and Programming) была начата в 1986 г. в г. Аахен (Германия). В 1997 г. центр координации разработки и технической поддержки пользователей переместился в Университет 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 др. наук из различных стран мира.

GAP является свободно распространяемой, открытой и расширяемой системой. Она распространяется в соответствии с GNU Public License.

GAP дает возможность производить вычисления с гигантскими целыми и рациональными числами, допустимые значения которых ограничены только объемом доступной памяти. Пользователю доступны различные комбинаторные функции, элементарные теоретико-числовые функции, разнообразные функции для работы с множествами и списками. Кроме того, система GAP позволяет решать различные задачи теории групп. Функции для работы с группами включают определение порядка группы, вычисление классов сопряженных элементов, центра и коммутанта группы, верхнего и нижнего центрального рядов, ряда коммутантов, силовских подгрупп, максимальных и нормальных подгрупп.

Среди других областей применения системы – теория графов и их автоморфизмов, теория кодирования, теория полугрупп, кристаллография, и многое другое. Информация о существующих разработках для применения GAP в той или проблемной области может быть найдена на сайте http://www.gap-system.org/.

Первые шаги в GAP. При успешном запуске GAP появится эмблема GAP. После нее будет напечатана дополнительная информация о версии системы и установленных компонентах, например:

```
 ┌─────────┐  GAP 4.11.0 of 29-Feb-2020
 │   GAP   │  https://www.gap-system.org
 └─────────┘  Architecture: x86_64-pc-cygwin-default64-kv7
Configuration:  gmp 6.1.2, GASMAN, readline
Loading the library and packages ...
Packages:   AClib 1.3.2, Alnuth 3.1.2,
      AtlasRep 2.1.0,
      AutoDoc 2019.09.04,
      AutPGrp 1.10.2, Browse 1.8.8,
      CaratInterface 2.3.3, CRISP 1.4.5,
      Cryst 4.1.23, CrystCat 1.1.9,
      CTblLib 1.2.2, FactInt 1.6.3,
      FGA 1.4.0, Forms 1.2.5,
```

GAPDoc 1.6.3, genss 1.6.6,
IO 4.7.0, IRREDSOL 1.4,
LAGUNA 3.9.3, orb 4.8.3,
Polenta 1.3.9, Polycyclic 2.15.1,
PrimGrp 3.4.0, RadiRoot 2.8,
recog 1.3.2, ResClasses 4.7.2,
SmallGrp 1.4.1, Sophus 1.24,
SpinSym 1.5.2, TomLib 1.2.9,
TransGrp 2.0.5, utils 0.69

Точный вид этого сообщения будет зависеть от набора установленных пакетов и их совместимости с операционной системой.

Приглашение системы (командная строка) имеет следующий вид:

```
gap>
```

Для выхода из системы применяется команда quit.

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

GAP можно использовать как простейший калькулятор:

```
gap> 5*(7/3+5/3);
20
gap> 2^10;
1024
```

Для удобства вычислений и написания формул в процессе диалога можно вводить переменные, например:

```
gap> a:=100;
100
gap> b:=25;
25
gap> (a+50)/b;
6
```

Пример. Определите последние шесть цифр числа Мерсенна $2^{82\,589\,933} - 1$, найденного в декабре 2018 г. в ходе проекта GIMPS (Great Internet Mersenne Prime Search) и являющегося на сегодня самым большим из известных науке простых чисел.

```
gap> a:=2^82589933-1;
<integer 148...591 (24862048 digits)>
gap> a mod 1000000;
902591
```

Лабораторная работа 1
«Основы работы с системой GAP»

Задание 1. Запустите систему GAP посредством ярлыка на рабочем столе или найдите каталог, в котором инсталлирована система GAP, и запустите файл gap.bat из каталога bin.

Простейшие вычисления можно выполнять, запуская систему из каталога таким образом, 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 на диске, где Вы имеете соответствующие права доступа, и скопировать туда файл gap.bat.

Задание 2. Выполните простейшие вычисления:

```
352/182;
2 *( 15 + 256) / 17;
2 ^ 64;
2 ^ 64 mod 100;
3 in [1,2,3];
2*2 >= 4;
```

Одна команда может занимать несколько строк, последняя из которых заканчивается точкой с запятой. Попробуйте ввести следующую многострочную команду:

```
155/4545 +
1234*5678 +
Sum([1..100]);
```

Задание 3. Скопируйте в буфер обмена команды, приведенные выше, а затем перейдите в окно GAP и вставьте их в командную строку (используйте сочетание клавиш Shift-Ins).

Задание 4. Одной из составных частей системы GAP является ее документация. Найдите описание функций Factorial и Sigma по ссылке https://www.gap-system.org/Manuals/doc/ref/chapInd.html. Скопируйте приведенные в документации примеры и выполните их в GAP.

Альтернативным вариантом использования документации является справка, которую можно вызвать прямо из командной строки GAP. 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 (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;
a:=2^(n +1) – 1;
IsPrime(a);
Factors(a);
x:= n + 10;
Factors (Factorial(x));
```

Теперь закройте файл протокола с помощью команды

```
LogTo();
```

и просмотрите его.

Задание 6. Автодополнение имен в системе GAP. Выясните, какие функции GAP содержат в названии фрагмент “Determinant” и “Gcd”. Для этого введите в командной строке (без точки с запятой)

```
gap> Determinant
gap> Gcd
```

и дважды нажмите *Tab*.

Язык программирования GAP

Символы и категории слов в GAP. GAP воспринимает следующие символы: цифры, буквы латинского алфавита (верхний и нижний регистры), пробел, символы табуляции и новой строки, а также специальные символы:

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

Составленные из символов слова относятся к следующим категориям:

1) ключевые слова – зарезервированные последовательности букв (перечень ключевых слов системы GAP можно получить с помощью команды GAPInfo.Keywords);

2) идентификаторы – последовательности букв (при этом регистр является существенным), цифр и символов подчеркивания, содержащие не менее одной буквы и не являющаяся ключевым словом;

3) строки – последовательности произвольных символов, заключенная в двойные кавычки;

4) целые числа (последовательности цифр);

5) символы-операторы и символы-разделители в соответствии со следующим списком

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

Выражения. 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). 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) унарные + и -;

3) *, / , mod;

4) +, -.

Логические операции: and (и), or (или), not (не). Эти операции выполняют с логическими переменными и константами. Результатом выполнения логической операции является значение логического типа.

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. Операторы сравнения имеют больший приоритет по сравнению с логическими операторами, но меньший по сравнению с арифметическими. Например, a*b = c and d интерпретируется как ((a*b)=c) and d).

Константы. Рассмотрим задание числовых и символьных констант на примерах.

```
gap> -7*3; 15/12;
-21
5/4
gap> 'a';
'a'
gap> "ab";
"ab"
```

Оператор присваивания имеет следующий формат:

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

```
gap> a:= 10;
10
gap> a * (a + 1);
110
```

Функции. GAP содержит более 4000 стандартных функций. Пример обращения к нескольким из них приведен ниже.

```
gap> Factorial(10);
3628800
gap> i:=5;;
gap> Print("i=",i," i^2=",i^2," i^3=", i^3, "\n" );
i=5 i^2=25 i^3=125
```

Заметим, что всякий раз, когда GAP возвращает значение, печатая его в следующей после команды строке, это значение присваивается переменной с именем last:

```
gap> 2^3;
8
gap> last^2;
64
```

Аналогичным образом определяются переменные last2 и last3.

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>.

Списки. Список в GAP является заключенным в квадратные скобки набором объектов, разделенных запятыми. Например, список из первых десяти простых чисел можно задать следующим образом:

```
gap> primes:=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29];
```

Затем к нему можно добавить следующие два простых числа:

```
gap> Append(primes, [31, 37]);
gap> primes;
```

Если добавляется только один элемент, это можно сделать и по-другому:

```
gap> Add(primes, 41);
gap> primes;
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41 ]
```

Указать отдельный элемент списка можно по его номеру в списке:

```
gap> primes[7];
17
```

Этот же механизм позволяет присвоить значение существующему или новому элементу списка

```
gap> primes[14]:= 43;
gap> primes;
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43 ]
```

При этом значение не обязательно должно присваиваться следующему элементу списка.

```
gap> primes[20]:=71;
71
gap> primes;
[2,3,5,7,11,13,17,19,23,29,31,37,41,43,,,,,,71]
gap> Length(primes);
20
```

Список может быть пустым

```
gap> lll:=[];
[ ]
gap> Length(lll);
0
gap> lll[1]:=5;
5
```

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

```
gap> Position(primes, 17);
7
gap> Position(primes, 20);
fail
```

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

```
gap> [1..100];;
gap> [5,10..100];;
gap> [100,95..5];;
gap> Length(last);
20
```

Для работы со списками используются также следующие встроенные функции:

List(list,func) возвращает новый список, в котором каждый *i*-й элемент является результатом выполнения функции func(list[i]).

```
gap> List( [1, 2, 3, 4, 5, 6, 1, 8, 9, 10], i -> i*2 );
[ 2, 4, 6, 8, 10, 12, 2, 16, 18, 20 ]
gap> List( [1, 2, 3, 4, 5, 6, 1, 8, 9, 10], IsPrime);
[ false, true, true, false, true, false, false, false, false, false ]
```

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

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

Строки. Строки являются частным случаем списков и печатаются без разделителей.

```
gap> text1:=['I',' ','l','o','v','e',' ','V','S','U','!'];
"I love VSU!"
gap> text2:="I love VSU!";
"I love VSU!"
gap> text1=text2;
true
```

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. Для каждого списка существует соответствующее ему множество, получаемое с помощью функции Set.

```
gap> list:=[5,10,15,5,10];
[ 5, 10, 15, 5, 10 ]
gap> set:=Set(list);
[ 5, 10, 15 ]
```

Проверить, является ли объект множеством, можно с помощью функции IsSet. 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. 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 функции IntersectSet, UniteSet и RemoveSet, но они не возвращают результат, а заменяют им первый аргумент.

Векторы. Вектор является не содержащим пробелов списком элементов.

```
gap> v:= [3, 6, 2, 5/2];
[ 3, 6, 2, 5/2 ]
gap> IsRowVector(v);
true
```

Векторы умножаются на скаляры, умножение двух векторов равной длины дает их скалярное произведение.

```
gap> 2*v;
[ 6, 12, 4, 5 ]
gap> v*1/3; # команда также может быть записана в виде v/3
[ 1, 2, 2/3, 5/6 ] # скалярное произведение v на себя
gap> v*v;
221/4
```

Матрицы. Матрица – список векторов одинаковой длины, не содержащий пробелов.

```
gap> m:= [[1, -1, 1],
> [2, 0, -1],
> [1, 1, 1]];
[ [ 1, -1, 1 ], [ 2, 0, -1 ], [ 1, 1, 1 ] ]
gap> m[2][1];
2
```

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

```
gap> m:= [[1, 2, 3, 4],
> [5, 6, 7, 8],
> [9,10,11,12]];;
gap> Display(m);
[ [  1,  2,  3,  4 ],
  [  5,  6,  7,  8 ],
  [  9, 10, 11, 12 ] ]
gap> [1, 0, 0, 0] * m;
[ 1, 2, 3, 4 ]
gap> [1, 0, 0] * m;
[ 1, 2, 3, 4 ]
gap> m * [1, 0, 0];
[ 1, 5, 9 ]
gap> m * [1, 0, 0, 0];
[ 1, 5, 9 ]
gap> m * [0, 1, 0, 0];
[ 2, 6, 10 ]
```

Подматрицы извлекаются или изменяются с помощью фигурных скобок. Первая пара скобок указывает выбранные строки, вторая – столбцы.

```
gap> sm := m{ [ 1, 2 ] }{ [ 3, 4 ] };
[ [ 3, 4 ], [ 7, 8 ] ]
gap> m{[1,2]}{[2]}:=[[0],[0]];
[ [ 0 ], [ 0 ] ]
gap> Display(m);
[ [  1,  0,  3,  4 ],
  [  5,  0,  7,  8 ],
  [  9, 10, 11, 12 ] ]
```

Лабораторная работа 2 «Списки»

Задание 1. Составьте примеры на использование каждой функции из приведенного ниже перечня.

Collected(list) – возвращает новый список new list, 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 из k элементов списка list (который может содержать одинаковые элементы несколько раз). Если *k* не указано, возвращаются все возможные комбинации, составленные из элементов списка list.

Divisorslnt(n) – возвращает список натуральных делителей целого числа *n*.

Factorslnt(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 $f(x)$ = true.

ForAll(list,x->f(x)) – проверяет, что для каждого элементам 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, … , a_n или целых чисел из списка list.

List(list,func) возвращает новый список, в котором каждый *i*-й элемент является результатом выполнения функции func(list[i]).

Length(list) определяет длину списка list.

a mod b – возвращает остаток от деления a на b.

Phi(n) – вычисляет $\varphi(n)$, т.е. количество чисел ряда 1, ..., n – 1 взаимно простых с *n*.

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 2. Задайте список с фамилиями студентов некоторой группы. Перечислите все возможные способы назначить двоих дежурных в данной группе.

Задание 3. Найдите количество способов разменять один рубль копейками. Используйте функцию RestrictedPartitions(n, set [,k]), которая возвращает множество всех (неупорядоченных) разбиений натурального числа *n* на *k* слагаемых (если задан аргумент *k*) или всевозможных (если он не задан), в которых слагаемые принадлежат множеству set.

Задание 4. Разложите на простые множители число *n*!, если

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 5. 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 6. 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 7. 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 8. 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 9. 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 10. 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 11. 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 12. 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»

Задание 1. Выполните примеры, приведенные ниже.

GAP содержит более 4000 стандартных функций. Кроме того, пользователь может вводить собственные функции. В общем случае задание функции осуществляется в следующем формате

```
function ( [ arg-ident {, arg-ident} ] )
        [ local loc-ident {, loc-ident} ; ]
        statements
        end
```

Наиболее просто это делается так.

```
gap> cubed:= x -> x^3;
function( x ) ... end
gap> cubed(5);
125
```

Рассмотрим задание функции, определяющей знак числа.

```
gap> sign:= function(n)
> if n < 0 then
> return -1;
> elif n = 0 then
> return 0;
> else
> return 1;
> fi;
> end;
function( n ) ... end
gap> sign(3); sign(0); sign(-3);
1
0
-1
```

Для работы с пользовательской функцией её задание можно сохранить в файле с расширением .g. В таком случае, для использования созданной функции она загружается в систему с помощью команды

Read (“*путь к файлу с расширением .g*”);

Определим функцию для вычисления суммы элементов побочной диагонали матрицы *A* третьего порядка, входным параметром которой является матрица *A*, а выходным – указанная сумма.

Создадим в рабочем каталоге файл lab3.g следующего содержания:

```
SumDiag:=function(A)
```

```
local s;
s := A[1][3] + A[2][2] + A[3][1];
return s;
end;
```

Теперь прочитаем его с помощью следующей команды (если система GAP запущена из другого каталога, то нужно будет указать полный путь к файлу):

```
gap> Read("lab3.g");
```

Теперь мы можем обращаться к функции SumDiag, например:

```
gap> M:=[[1,2,3],[-1,-2,-3],[1,1,1]];;
gap> SumDiag(M);
2
```

Замечание. При разработке функций обращайте внимание на следующее:

• отсутствие точки с запятой в первой строке;

• обязательность точки с запятой после команды end;

• указание локальных переменных с помощью команды local;

• сообщение об ошибке, если имя Вашей функции совпадет с именем одной из библиотечных функций GAP.

Задание 2. Разработайте функцию, входным параметром которой являются координаты двух точек плоскости A и B, а выходным – координаты середины отрезка AB.

Задание 3. Разработайте функцию для транспонирования матрицы третьего порядка непосредственно по определению транспонирования матриц (т.е. не используя имеющуюся в GAP операцию транспонирования).

Задание 4. Разработайте функцию для вычисления определителя матрицы A третьего порядка.

Задание 5. Разработайте функцию для вычисления суммы элементов побочной диагонали матрицы A любого порядка. Функция должна проверять является ли заданная матрица квадратной. Для определения количества строк и столбцов матрицы A можно воспользоваться следующими командами

```
nrows := Length( A ); # количество строк
ncols := Length( A[1] ); # количество столбцов
```

Условный оператор и операторы циклов в GAP

Команда IF. Формат:

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

При этом часть else может отсутствовать, частей elif может быть произвольное количество или ни одной.

Пример. Разработайте функцию, которая выводит решение уравнения $|x| = a$, если входным параметром данной функции является значение a.

Создадим в рабочем каталоге файл lab4.g следующего содержания:

```
EqModul:=function(a)
if a<0 then Print("уравнение |x|=",a, " не имеет решений");
elif a=0 then Print("уравнение |x|=",a, " имеет один корень x=0");
else Print("уравнение |x|=",a," имеет два корня x1=",a," и ","x2=",-a);
fi;
end;
```

Теперь прочитаем его с помощью следующей команды (если система GAP запущена из другого каталога, то нужно будет указать полный путь к файлу):

```
gap> Read("lab4.g");
```

Решим несколько уравнений с помощью полученной функции

```
gap> EqModul(-5);
gap> EqModul(0);
gap> EqModul(5);
```

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. 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, пока <условие> ложно. При этом сначала выполняются команды, а затем проверяется условие. Таким образом, при любом значении условия <последовательность команд> выполняется, по крайней мере, один раз.

Пример. 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;

Пример. Найдите порядок группы, порожденной перестановками ((l,2,3,4,5),(l,2)(3,4)(5,6)) и вычислите сумму порядков всех её элементов

```
gap> g := Group((1,2,3,4,5),(1,2)(3,4)(5,6));;
gap> count := 0;; sumord := 0;;
gap> for x in g do
> count := count + 1;
> sumord := sumord + Order(x);
> od;
gap> count;
120
gap> sumord;
471
```

Пример. Построим таблицу истинности для логического оператора and. Сначала зададим список *xy*, состоящий из всевозможных комбинаций true и false. Воспользуемся для этого функцией Tuples(set, k), которая возвращает размещения с повторениями, т.е. всевозможные упорядоченные наборы из *k* элементов исходного множества *set*.

```
gap> xy:=Tuples([true, false], 2);
```

[[true, true], [true, false], [false, true], [false, false]]

Теперь будем перебирать все пары из этого списка с помощью цикла for. Для каждой пары (x,y) выведем на печать значение высказываний "x и y", "x или y".

```
gap> for a in xy do
> Print(a, "  ", a[1] and a[2], "  ", a[1] or a[2], "\n");
> od;
[ true, true ]  true  true
[ true, false ]  false  true
[ false, true ]  false  true
[ false, false ]  false  false
```

Лабораторная работа 4
«Ветвящиеся и циклические программы»

Задание 1. Разработайте функцию, которая проверяет является ли заданное значение *a*, корнем многочлена. Выходной параметр – значение true или false.

Примечание. 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");
x
```

После этого многочлены можно задавать привычным образом:

```
gap> f:=x^3+5*x^2-7*x+1;
x^3+5*x^2-7*x+1
```

Функция Value(f,a) вычисляет значение многочлена *f* при заданном значении переменной $x = a$.

```
gap> Value(f, 0);
1
```

Задание 2. Разработайте функцию, которая определяет, являются ли коэффициенты заданного многочлена взаимно простыми, и в случае положительного ответа возвращает исходный многочлен, а в случае отрицательного – исходный многочлен, разделенный на НОД своих коэффициентов. При выполнении задания воспользуйтесь функцией CoefficientsOfUnivariatePolynomial(f), которая возвращает список коэффициентов многочлена *f*.

Задание 3. Разработайте функцию, которая определяет, равен ли единице старший коэффициент заданного многочлена *f*, и в случае положительного ответа возвращает исходный многочлен, а в случае отрицательного – многочлен, полученный из многочлена *f* делением его на свой старший коэффициент.

Задание 4. Разработайте функцию, которая определяет порядок заданной подстановки по определению, вычисляя минимальную степень, в которую нужно ее возвести, чтобы получить тождественную подстановку (т.е. не используя имеющуюся в GAP функцию для вычисления порядка).

Замечание. Имя функции должно быть выбрано таким образом, чтобы оно не совпадало с именем имеющейся в GAP стандартной функции Order, которая определяет порядок элемента группы. В противном случае, при ее

вводе будет получено сообщение об ошибке, т.к. библиотечные функции защищены от переопределения пользователем.

Задание 5. Разработать функцию, которая для заданной подстановки s определяет минимальное натуральное число k, такое что s^k коммутирует с заданной подстановкой t.

Задание 6. Постройте таблицы истинности для импликации и эквиваленции.

Задание 7. С помощью таблицы истинности докажите один из законов дистрибутивности алгебры высказываний.

Элементы теории групп в GAP

Напомним основные определения теории групп.

Определение. 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 $a^{-1}a = a\,a^{-1} = e$.

Определение. 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.

Примеры.

1. Множества целых чисел $\mathbb{Z}$, рациональных чисел $\mathbb{Q}$, действительных чисел $\mathbb{R}$, комплексных чисел $\mathbb{C}$ с операцией сложения – абелевы группы.

2. Множество натуральных чисел $\mathbb{N}$ со сложением не является группой, т.к. в $\mathbb{N}$ нет нулевого и противоположных элементов. Однако $\mathbb{N}$ со сложением – коммутативная полугруппа.

3. Множество $\{-1,1\}$ с умножением – 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 $\mathbb{C}^* = \mathbb{C}\backslash\{0\}$, $\mathbb{R}^* = \mathbb{R}\backslash\{0\}$, $\mathbb{Q}^* = \mathbb{Q}\backslash\{0\}$, то $\mathbb{C}^*$, $\mathbb{R}^*$ и $\mathbb{Q}^*$ с умножением являются абелевыми группами.

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 a – фиксированный элемент произвольной группы G. Обозначим $\langle a\rangle = \{a^m | m \in Z\}$ – множество всевозможных целых степеней элемента a, где $a^0 = e$, $a^n = \underbrace{a \cdot \ldots \cdot a}_{n\ раз}$, $a^{-n} = (a^{-1})^n$. Тогда $\langle a\rangle$ – абелева группа, которая называется циклической группой, порожденной элементом a.

Пусть a – произвольный элемент группы G. Рассмотрим два возможных случая.

1. Все степени элемента a различны, т.е. $a^m \neq a^n$ для всех целых $m \neq n$. В этом случае говорят, что элемент a имеет бесконечный порядок.

2. Имеются совпадения $a^m = a^n$ при $m \neq n$. Если, например, $m > n$, то $m - n > 0$ и $a^{m-n} = e$, т.е. существуют натуральные степени элемента a, равные единичному элементу. Наименьшее натуральное число k, при котором $a^k = e$, называют порядком элемента a и пишут $|a| = k$.

Теорема. Пусть элемент $a \in G$ имеет конечный порядок k. Тогда $\langle a \rangle = \{e, a, a^2, \ldots, a^{k-1}\}$. Кроме того, $a^m = e$ в точности тогда, когда k делит m.

Пусть Ω – конечное множество из n элементов. Поскольку природа его элементов для нас не существенна, удобно считать, что $\Omega = \{1, 2, \ldots, n\}$.

Определение. Всякая биекция, то есть взаимно однозначное отображение Ω в себя, называется подстановкой на Ω.

Подстановку f изображают в виде двустрочной таблицы:

$$f = \begin{pmatrix} 1 & 2 & \ldots & n \\ f(1) & f(2) & \ldots & f(n) \end{pmatrix}.$$

В этой таблице каждый i-й столбец указывает, в какой элемент $f(i)$ преобразуется элемент i, $1 \le i \le n$.

Подстановки перемножаются в соответствии с общим правилом композиции отображений: $(gf)(i) = g(f(i))$. Очевидно, что тождественная подстановка $e = \begin{pmatrix} 1 & 2 & \ldots & n \\ 1 & 2 & \ldots & n \end{pmatrix}$ является нейтральным элементом относительно композиции подстановок. Как известно, композиция отображений является ассоциативной операцией, поэтому и композиция подстановок ассоциативна. Каждая подстановка – обратимая операция. Чтобы найти для подстановки f обратную подстановку f^{-1} достаточно в таблице $f = \begin{pmatrix} 1 & 2 & \ldots & n \\ f(1) & f(2) & \ldots & f(n) \end{pmatrix}$ переставить строки местами, а затем столбцы упорядочить по возрастанию элементов первой строки. Таким образом, подстановки на Ω образуют группу относительно операции композиции отображений – умножения подстановок. Ее называют симметрической группой степени n и обозначают через S_n. При $n > 3$ эта группа неабелева.

Теорема. Порядок группы S_n равен $n!$.

Рассмотрим подстановку, в которой элементы переставляются по циклу:

$$i_1 \to i_2 \to \cdots \to i_{k-1} \to i_k \to i_1.$$

Такой цикл записывают в виде $(i_1, i_2, \ldots, i_{k-1}, i_k)$, но можно также начинать запись с любого другого элемента из цикла: $(i_t, i_{t+1}, \ldots, i_k, i_1, i_2, \ldots, i_{t-1})$. Любую подстановку можно разложить в произведение циклов, которые попарно не имеют общих элементов.

Пример. Запишем в виде циклов все подстановки из четырех элементов: тождественная подстановка e, циклы длины два – (1 2), (1 3), (1 4), (2 3), (2 4), (3 4), циклы длины три – (1 2 3), (3 2 1), (1 2 4), (4 2 1), (1 3 4), (4 3 1), (2 3 4), (4 3 2), циклы длины четыре – (1 2 3 4), (4 3 2 1), (1 2 4 3), (3 4 2 1), (1 3 2 4), (4 2 3 1) и пары циклов длины два – (1 2)(3 4), (1 3)(2 4) и (1 4)(2 3).

Циклы длины два называют транспозициями.

Теорема. Каждая подстановка $f \in S_n$ является произведением транспозиций.

Пусть задана подстановка $f = \begin{pmatrix} 1 & 2 & \dots & n \\ f(1) & f(2) & \dots & f(n) \end{pmatrix}$. Говорят, что пара $\{i, k\}$ образует инверсию по отношению к подстановке f, если $i < k$, но при этом $f(i) > f(k)$. Подстановка называется четной, если она содержит четное число инверсий; подстановка называется нечетной, если она содержит нечетное число инверсий. Тождественная подстановка является четной. Четные подстановки образуют конечную группу A_n порядка $n!/2$, которую называют знакопеременной группой степени n.

Определение. Подмножество H группы G называется подгруппой, если H – группа относительно той же операции, которая определена на G. Запись $H < G$ означает, что H – подгруппа группы G, а $H \le G$, что H – собственная подгруппа группы G, т.е. $H < G$ и $H \ne G$.

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.

Примеры подгрупп (относительно умножения):

1. $\{-1,1\} < \mathbb{Q}^* < \mathbb{R}^* < \mathbb{C}^*$.
2. $A_n < S_n$

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

Теорема (Лагранжа). Порядок конечной группы делится на порядок любой ее подгруппы.

Определение. 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 $<M>$.

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

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

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

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.

Пусть p – простое число. Конечная группа, порядок которой есть степень простого числа p, называется p-группой. Максимальная p-подгруппа называется силовской p-подгруппой.

Теорема (Силова). Пусть конечная группа G имеет порядок $p^m s$, где p – простое число, $m \in \mathbb{N}$, p не делит s. Тогда справедливы следующие утверждения:

1) в группе G существует силовская p-подгруппа и ее порядок равен p^m;

2) каждая p-подгруппа содержится в некоторой силовской p-подгруппе;

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

4) число силовских p-подгрупп группы G сравнимо с единицей по модулю p и делит s.

Пусть $\mathbb{P}$ – множество всех простых чисел, а π – некоторое множество простых чисел, то есть $\pi \subseteq \mathbb{P}$. Дополнение к π во множестве $\mathbb{P}$ обозначим через π' , то есть $\pi' = \mathbb{P} \backslash \pi$. Будем использовать функцию $\pi(m)$ – множество всех простых чисел, делящих натуральное число m. Зафиксируем множество простых чисел π. Если $\pi(m) \subseteq \pi$, то число m называется π-числом.

Подгруппа H группы G называется π-подгруппой, если $|H|$ есть π-число. Подгруппа H называется холловой π-подгруппой, если $|H|$ есть π-число, а индекс $| G : H |$ есть π'-число.

Лабораторная работа 5
«Группы, свойства элементов группы, подгруппы»

Задание 1. Разберите и выполните примеры 1-4.

Рассмотрим некоторые функции, которые в текущей версии GAP используются для работы с группами.

Group(gl, g2, ..., gN) группа, порожденная элементами gl, g2, ..., gN;

GroupByGenerators(gens) группа, порожденная элементами из списка gens;

GeneratorsOfGroup(G) возвращает список порождающих элементов группы G;

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;

Пример 1.

```
gap> G:=Group((1,2),(1,2,4));
gap> AsList(G); # список элементов группы
gap> S:=SymmetricGroup(5);
gap> GeneratorsOfGroup(S);
```

Библиотека групп малых порядков в системе GAP содержит все конечные группы, порядок которых не превышает 2000, за исключением групп порядка 1 024. Каждая группа из библиотеки имеет свой номер, обозначающий ее тип изоморфизма. Этот номер имеет вид $[n_1, n_2]$, где n_1 – порядок группы, n_2 – её номер в каталоге групп порядка n_1. Группу, имеющую тип изоморфизма $[n_1, n_2]$, можно вызвать из библиотеки с помощью функции SmallGroup.

Пример 2.

```
gap> S:= SmallGroup (24,12);
```

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:

Пример 3.

```
gap> S:=SymmetricGroup(4);
gap> IdGroup(S);
```

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:

Пример 4.

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

Задание 2. Найдите количество всех нециклических групп порядка 32. Для решения задания используйте функцию IsCyclic.

Задание 3. Разработайте функцию, которая возвращает множество элементов заданной группы, имеющих порядок, равный заданному числу k. Указание: используйте функцию Order

Задание 4. Найдите все 2-элементные множества, порождающие симметрическую группу S_3.

Задание 5. Пусть A и B – подгруппы симметрической группы S_n, порожденные множествами gens1 и gens2. Разработайте функцию, которая проверяет является ли подгруппой произведение AB.

Задание 6. Разберите и выполните примеры 5-6.

Функция SylowSubgroup(G,p) возвращает силовскую p-подгруппа конечной группы G.

Пример 5. Для каждого из простых делителей p вычислим порядок силовской p-подгруппы группы S_8.

```
gap> S:=SymmetricGroup(8);
gap> Set(Factors(Size(S)));
gap> for p in last do
> Print(p, " : ", Size(SylowSubgroup(S,p)),"\n");
> od;
```

Пример 6. В каких силовских 2-подгруппах группы S_4 содержится подстановка (1, 3, 2, 4)?

Сначала зададим исходную группу

```
gap> S:=SymmetricGroup(4);
```

Вычислим ее силовские 2-подгруппы. Поскольку по теореме Силова все силовские p-подгруппы сопряжены, то функция SylowSubgroup возвращает только одну подгруппу, которая является представителем некоторого класса сопряженных подгрупп.

```
gap> P2:= SylowSubgroup(S,2);
```

Для того, чтобы получить остальные подгруппы из этого класса, сначала нужно создать класс сопряженных подгрупп данной группы с

заданным представителем (посредством функции ConjugacyClassSubgroups), а затем получить список содержащихся в нем подгрупп.

```
gap> c2:= ConjugacyClassSubgroups(S,P2);
gap> l2:= AsList(c2);
```

Теперь мы можем получить из него список подгрупп, которые содержат указанную подстановку:

```
gap> Filtered( l2, g -> (1,3,2,4) in g );
```

Задание 7. Разработайте функцию, которая для заданной группы возвращает список простых делителей порядка группы с указанием количества соответствующих силовских p-подгрупп.

Задание 8. Пусть $\pi(G)$ – множество всех простых делителей порядка группы G. С помощью функции HallSubgroup(G,pi) определите для каких значений pi $\subseteq \pi(G)$ в группе $G = S_8$ существуют холловы pi-подгруппы.

Литература

1. Компьютерная алгебра : учеб.-метод. комплекс для студентов специальности 1-02 05 01 «Математика и информатика» физико-математического факультета / Д.В. Грицук, А.А. Трофимук ; Учреждение образования «Брестский государственный университет имени А.С. Пушкина», каф. АГ и ММ – Брест : электронное издание БрГУ, 2018. – 270 с.

2. 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. – Дата доступа: 01.09.2021.

3. Монахов, В.С. Введение в теорию конечных групп и их классов / В. С. Монахов. – Минск : Выш. шк., 2006. – 207 с.

Учебное издание

КОМПЬЮТЕРНАЯ АЛГЕБРА

Методические рекомендации

Составитель
ВИТЬКО Елена Анатольевна

Технический редактор *Г.В. Разбоева*
Компьютерный дизайн *Л.Р. Жигунова*

Подписано в печать 2021. Формат 60x84 $^{1}/_{16}$. Бумага офсетная.
Усл. печ. л. 1,98. Уч.-изд. л. 1,08. Тираж экз. Заказ .

Издатель и полиграфическое исполнение – учреждение образования
«Витебский государственный университет имени П.М. Машерова».

Свидетельство о государственной регистрации в качестве издателя,
изготовителя, распространителя печатных изданий
№ 1/255 от 31.03.2014.

Отпечатано на ризографе учреждения образования
«Витебский государственный университет имени П.М. Машерова».
210038, г. Витебск, Московский проспект, 33.