This is that part of the CHEVIE manual from Version 2.1 which describes the Maple part of Chevie. Since this part of Chevie has not changed in Version 3.1, it is still valid.

CHEVIE—Generic Character Tables of Finite Groups of Lie Type, Hecke Algebras and Weyl Groups[1]

Version 2.1

Meinolf Geck, Götz Pfeiffer

Lehrstuhl D für Mathematik, RWTH Aachen
Templergraben 64, D–52062 Aachen

Gerhard Hiss, Frank Lübeck, Gunter Malle

IWR der Universität Heidelberg
Im Neuenheimer Feld 368, D–69120 Heidelberg

August 26, 1993

[1]Work on CHEVIE is financially supported by the Deutsche Forschungsgemeinschaft as part of the Forschungsschwerpunkt *Algorithmische Zahlentheorie und Algebra*

Preface

CHEVIE is a package based on the computer algebra systems GAP [20] and MAPLE [6]. It consists of a library of data and programs for dealing with generic character tables of finite groups of Lie type and associated structures such as finite Weyl groups and Hecke algebras.

Preliminary versions of this system have already proved to be useful in various applications in constructive Galois theory, investigations on the subgroup structure and the modular character theory of finite groups of Lie type. One of the motivations for us was to provide an appropriate platform and environment for collecting known results, for working out new results in a systematic manner, and for performing experiments on substantial examples. Moreover, we found it most convenient to have electronic access to large tables previously available only in printed form. Unfortunately, working with such printed tables often is quite cumbersome due to occasional inaccuracies which are difficult to spot. Note that, e. g., a printed version of the generic character table of $CSp_6(q)$, q odd, which is part of CHEVIE, would require several hundred pages. Thus, we also hope to provide a means for putting into print results of this type, with the risk of misprints minimized.

One of our main inspirations was the idea underlying the famous Cambridge Atlas of finite groups [7] which has become an important tool for investigating finite groups and their representations. While the Atlas contains the character tables of individual groups (like $GL_2(5)$) our objective was to give one generic character table for all groups in a series (like $GL_2(q)$, q any prime power). First examples of such generic tables were already given by I. Schur (for $SL_2(q)$, see [21]) at the beginning of the century. J. A. Green [13] completely described the irreducible characters of the finite general linear groups. Explicit tables for other groups of small rank were determined by B. Srinivasan (for $Sp_4(q)$, see [22]) and B. Chang, R. Ree (for $G_2(q)$, see [5]). Using the description of finite groups of Lie type as fixed point sets of connected reductive groups under a Frobenius map, P. Deligne and G. Lusztig in [8] introduced new methods from algebraic geometry and ℓ-adic (intersection) cohomology in the study of irreducible representations of these groups. This enabled Lusztig finally to give a classification of the irreducible characters (see [16]) and a general procedure for constructing the irreducible characters in an algorithmic fashion—at least in principle, and up to scalar multiples (see [17] and the references there). Nevertheless, the explicit construction of a particular table still requires a huge amount of work and detailed knowledge.

Finite Weyl groups, associated Hecke algebras and their representations play an important role in these theories. Certain specializations of Hecke algebras arise naturally as endomorphism rings of representations obtained by Harish-Chandra induction from Levi subgroups. However, results on representations of Hecke algebras also have an independent interest as well as applications in other areas of mathematics (e.g., in V. F. R. Jones' construction of a two-variable polynomial invariant of knots and links, see [15]). Much of the first work on Hecke algebras was centered around the determination of the so-called generic degrees (see [1]). D. Kazhdan and G. Lusztig introduced the new concepts of left cells and their geometric interpretation in terms of intersection cohomology of Schubert varieties. Recently, general concepts in the representation theory of Hecke algebras have been developed, and explicit results on character tables and decompositions numbers were computed (see [12] and [11]).

We shall now give a brief sketch of the content and structure of CHEVIE, Version 2.0.

- Generic ordinary character tables for all series of finite groups of Lie type of small rank.

- Tables of Green polynomials (for exceptional and classical groups of low rank, and disconnected groups), and a program for calculating the Green polynomials for groups of type A.

- Programs for working with these tables (scalar products, tensor products, structure constants), constructing new tables and viewing respectively printing them in TEX format.

- Character tables of Hecke algebras for all types of rank at most 7, and recursive algorithms for computing them for types A_n and B_n (the formulae for type B and rank larger than 7 give the correct result only conjecturally). Tables and programs for the Weyl groups themselves are already available through GAP.

- Programs for computing Kazhdan–Lusztig polynomials, left cells and the corresponding representations for Weyl groups (effective for rank ≤ 4).

Our idea of using computer algebra systems such as MAPLE for performing symbolic calculations with the characters of a whole series of groups of Lie type dates back in 1986. We determined decomposition numbers for the groups $G_2(q)$ [14] and $SU_3(q)$ [10] with the assistance of MAPLE. In another direction, problems of constructive Galois theory motivated the computation of the unipotent characters of the Ree groups ${}^2F_4(q^2)$ [18] and also Green functions for disconnected groups [19]. These calculations involved the tedious work of typing existing tables into the computer, checking and correcting them. The amount of data became so large that we were forced to think about a systematic approach. At about 1990, a preliminary version of GAP was available. We used it as a platform for programs dealing with finite Weyl groups and Hecke algebras. (These are also available through GAP, Version 3, Release 1). We then decided to combine the various tables, GAP and MAPLE procedures into a single system with well defined data structures: Version 1 of CHEVIE (unpublished).

CHEVIE is currently used by a PhD-student for the construction of the unipotent characters of $F_4(q)$. One of us (FL) is extending CHEVIE by programs which allow the automatic calculation of the head of a generic character table, and also of the Deligne–Lusztig characters $R_{T,1}$. One of the applications of this will be the construction of the unipotent characters of $E_6(q)$.

We close by inviting the users of CHEVIE to communicate their suggestions, criticisms and own contributions to the authors.

Aachen, Heidelberg
October 1993

M. G., G. H., F. L., G. M., G. P.

Preface to Version 2 Release 1

Release 1 of Version 2 of CHEVIE does not differ substantially from Release 0. Some bugs have been fixed and some tables of unipotent characters have been added.

The MAPLE part of Release 0 is written for MAPLE V.2 and does not run with MAPLE V.3. We now provide two versions of CHEVIE, one for MAPLE V.2 and older releases of MAPLE and one for MAPLE V.3. Due to changes in the syntax of MAPLE it is not possible to have a single version running for all MAPLE releases.

Aachen, Heidelberg
August 1994

M. G., G. H., F. L., G. M., G. P.

Acknowledgements

It is a pleasure to acknowledge the help and support of many people and institutions. First of all we thank the Deutsche Forschungsgemeinschaft for their financial support of the CHEVIE project.

The concepts underlying the CHEVIE manual, the conventions for the TEX-source files containing it and the additional TEX macros needed to typeset it are taken from GAP. We thank Joachim Neubüser and the Aachen GAP group for the permission to use all this but also for much more support and encouragement.

The following students have contributed programs, tables and ideas to CHEVIE: Marianne Feltgen, Ulrich Porsch and Elmar Wings.

Finally we would like to thank Raphael Nauheim for his constant and friendly help with the subtleties of MAPLE but in particular with the more advanced features of TEX.

Contents

Chapter 1

Characters and Classes of Finite Reductive Group

This chapter is very sketchy. We plan to extend it in a later edition.

1.1 Generic Finite Reductive Groups

A general reference for finite reductive groups and their representation theory are the books by R. W. Carter [4] and F. Digne and J. Michel [9].

The concept of a *generic finite reductive group* was introduced by Broué and Malle in [2]. Here, we follow M. Broué, G. Malle and J. Michel [3]. A generic finite reductive group is also called a *series of finite reductive groups*. Such a series of finite reductive groups is the collection of all finite reductive groups associated to a *complete root datum*.

A complete root datum or a generic finite reductive group is a pair $\mathbb{G} = (\Gamma_{\mathbb{G}}, W_{\mathbb{G}}\phi)$, where $\Gamma_{\mathbb{G}}$ is a root datum, $W_{\mathbb{G}}$ is its Weyl group and ϕ is an automorphism of finite order of $\Gamma_{\mathbb{G}}$.

Given a root datum Γ and a prime number p, there exists a pair $(\mathbf{G}, \mathbf{T})$, where $\mathbf{G}$ is a connected reductive group defined over $\bar{\mathbb{F}}_p$, and $\mathbf{T}$ is a maximal torus of $\mathbf{G}$ such that Γ is the root datum associated to the pair $(\mathbf{G}, \mathbf{T})$. Given a complete root datum $\mathbb{G} = (\Gamma_{\mathbb{G}}, W_{\mathbb{G}}\phi)$, a prime number p, a power q of p and an element ϕ' in the coset $W_{\mathbb{G}}\phi$, there is an associated Frobenius morphism $F : \mathbf{G} \to \mathbf{G}$, such that $\mathbf{T}$ is F-stable, and thus an associated finite reductive group, denoted by $\mathbf{G}^F = \mathbb{G}(q)$. The collection of all groups $\mathbb{G}(q)$, $q = p^f$, $f \in \mathbb{N}$, p a prime number, is called a *series of finite groups of Lie type*.

The definition of the Suzuki and Ree groups is slightly more complicated. Let us content ourselves with the remark that the Suzuki groups $Sz(q^2) = {}^2B_2(q^2)$, $q^2 = 2^{2n+1}$, and the Ree groups ${}^2G_2(q^2)$, $q^2 = 3^{2n+1}$, respectively ${}^2F_4(q^2)$, $q^2 = 2^{2n+1}$, each form a series in its own right.

1.2 Semisimple Section Types

Given a generic finite reductive group $\mathbb{G}$, a prime p and a power q of p with corresponding algebraic group $\mathbf{G}$, we introduce an equivalence relation on the set of conjugacy classes of $G = \mathbb{G}(q)$ as follows. Let $g \in G$ with Jordan decomposition $g = su = us$, where s and u denote the semisimple respectively unipotent part of g. If g' is another element of G with Jordan decomposition $g' = s'u' = u's'$ we say that g and g' have the same *semisimple section type*, if and only if there exists $h \in G$ such that $C_{\mathbf{G}}(s)^h = C_{\mathbf{G}}(s')$. This is clearly an equivalence relation on G and the equivalence classes are unions of conjugacy classes. The set of semisimple conjugacy classes of a semisimple section type is called a *semisimple class type*.

1.3 Lusztig Series Types

To define the Lusztig series types, we have to recall Lusztig's Jordan decomposition of characters. For simplicity we assume that the underlying algebraic group $\mathbf{G}$ has a connected centre. Then every ordinary irreducible character of G is parametrized by a pair (s^*, λ), where s^* is a semisimple element of the dual group $G^* = \mathbf{G}^{*F^*}$, and λ is a unipotent character of the F-fixed points of $C_{\mathbf{G}^*}(s^*)^*$. We say that two irreducible characters corresponding to (s^*, λ) respectively $({s'}^*, \lambda')$ belong to the same Lusztig series type, if s^* and ${s'}^*$ have the same semisimple class type. The set of irreducible semisimple characters in a Lusztig series type is called a *semisimple character type*.

1.4 Generic Character Tables

Let s be a semisimple element of G and let $\mathbf{C} = C_{\mathbf{G}}(s)$ denote its centralizer in $\mathbf{G}$. Let $u_0, u_1, \ldots, u_r$ denote representatives of the unipotent conjugacy classes of $\mathbf{C}^F = C_G(s)$. The generic character table has $r+1$ columns corresponding to the semisimple class type of s, one column for each pair (s, u_i). Let $s_1, s_2, \ldots, s_t$ denote representatives for the semisimple conjugacy classes of G which have the same section type as s. If u is one of the u_i, the column of the generic character table corresponding to the pair (s, u) represents the t columns of the conventional ordinary character table of G with the representatives $s_1u, s_2u, \ldots, s_tu$.

A similar convention is used to introduce the rows of the generic character table.

Chapter 2

Conventions for Generic Character Tables

2.1 Tables and Names

table: In the following, the word *table* is used for a generic character table or a table of Green functions. Various names are associated to a table. The word *character* denotes a character type of a generic character table and also a Green function of a table of Green functions. A similar convention is used with the word *class*.

file name: All information for a table is stored on a single file. Its name is called the *file name* of a table. The file name should indicate its contents. So, for example, there is a file `chevie/tables/A2/GL3` containing the generic character table for the Chevalley group $GL_3(q)$, and there is a file `chevie/greenfunctions/A2/GL3`, containing the table of Green functions for $GL_3(q)$. In some cases, a table has more than one file name. For example, the file names `2G2` and `ree` are associated to the generic character tables for the Ree groups of type ${}^2G_2$.

table name: The character values of a table are stored in a MAPLE array. The name of this array is called the *table name*. For most of the generic character tables the file name and table name are identical. The table name can be changed in a CHEVIE session. In fact, after reading a table from a file, CHEVIE assigns the name `g` to the table. This assignment is printed onto the screen as `g := 'old_table_name'`, so that the `old_table_name` can be read off.

table number: Procedures for summing over all characters inside a given character type are part of a generic character table. Since CHEVIE can work with several tables at the same time, these procedures have to be linked to the individual tables. This is done via the *table number*, a MAPLE string identifying the table uniquely. The table number is a concatenation of three substrings:

```
add.a.bc
```

The first substring, `add`, is a symbol for the Dynkin diagram and an automorphism thereof, for example `G2`, `2A2` or `3D4`. The second substring, `a`, a decimal digit, indicates the isogeny type and the structure of the centre of the underlying algebraic group. The digit `0` is used for a standard type group (yet to be defined precisely). The digit `1` is used for simple adjoint groups, and `2` is used for simple simply connected groups.

The third substring, `bc`, consists of two decimal digits. It is used to distinguish between tables for the same Dynkin, isogeny and central type, but different congruence classes of `q`. The two digits stand for the smallest value of `q` in the congruence class (written in the decimal system with leading `0`).

Examples: `G2002` is the table number of the generic character table for the groups $G_2(q)$ with even q congruent to 2 modulo 3. Its file name is `G2.02`.

`G2007` is the table number of the generic character table for the groups $G_2(q)$ with odd q congruent to 1 modulo 3. Its file name is `G2.11`.

`2A2202` is the table number for $SU_3(q)$ for q congruent to 2 modulo 3. Its file name is `SU3.2`.

2.2 The Array of a Table

Let *cht* respectively *clt* denote the number of character respectively class types of a table T. Then the MAPLE array `T` containing T has $3 + cht + r$ rows and $2 + clt + c$ columns, where r and c are non-negative integers. The numbering of the rows of `T` starts with -2, the numbering of the columns with -1:

```
T := array(-2 .. cht + r, -1 .. clt + c, [])
```

The value of character `i` on class `j` is stored on position `[i,j]` of `T`. Additional information is stored on rows `-2 .. 0` and columns `-1 .. 0` of `T` according to the following list.

Rows	Columns	Contents
-2	-1	TeX string for name of group
	0	Table Number
	1	Order of group
	2	Number of character types stored on `T` (cht)
	3	Number of rows available on `T` ($cht + r$)
	4	Number of class types stored on `T` (clt)
	5	Number of columns available on `T` ($clt + c$)
-1	-1	Not defined yet
	0	Not defined yet
	$1 \ldots clt$	Information lists for class types
0	-1	Not defined yet
	0	Not defined yet
	$1 \ldots clt$	Lengths of conjugacy classes
$1 \ldots cht$	-1	Information lists for character types
	0	Character degrees
	$1 \ldots clt$	Character values

Note that for each table, $clt + c$ must be at least 5.

The information concerning the names of the parameters for character respectively class types and the ranges of these parameters is not stored on `T`. There are two 1-dimensional arrays

```
Char.T.Parameter := array(-2 .. cht + r,[]);

Klassen.T.Parameter := array(-1 .. clt + c,[]);
```

They contain, on position i, the information for the character type respectively class type stored on row i respectively column i of `T`. The entries are lists, which are used by the functions `PrintCharParam` respectively `PrintClassParam`.

In order to prevent unwanted re-assignment, the names of the parameters for the characters and conjugacy classes have the form `xX`, where `x` can be any non-capital letter from the alphabet, and `X` is its capitalized form. So, for example, `kK` is a single parameter, not the product of the parameters `k` and `K`. In `PrintToTeX`, however, `kK` is printed as k.

2.3 Summation Procedures

For each character respectively class type, there is a MAPLE procedure for summing over all the characters respectively classes in the given type. Let tn stand for the table number.

The name for the class summation procedure for class type cl is

`Klassen.`tn`.Summe.`cl

The name for the character summation procedure for character type ch is

`Char.`tn`.Summe.`ch

2.4 Exceptions

In various functions, such as e.g. `Scalar`, CHEVIE returns a message `Possible Exceptions`, which will now be explained.

CHEVIE does generic calculations, which may not be valid for some special values of the parameters. The possible exceptions are printed as a set of lists of the form

`[ expr, poly ]`,

where `expr` is an expression in the parameters and `poly` a polynomial in `q`.

The calculations are correct for all parameter combinations (inside the allowed range of the parameters), except possibly for those, where `poly` divides `expr`.

```
Possible Exceptions:     {[uU-vV, q+1], [uU-wW, q+1], [vV-wW, q+1]}
```

The calculations leading to the above message are valid for all triples `(uU,vV,wW)` of parameters except possibly for those where `q + 1` divides one of `uU - vV` or `uU - wW` or `vV - wW`.

2.5 GEW

Here, we shortly explain the CHEVIE notation for irrationalities, in particular for generic roots of unity.

`EW` respectively `GEW` stand for *root of unity* respectively *generic root of unity*.

`EWi` $:= \exp(2\pi\sqrt{-1}/i)$, i a positive integer,

`GEWZi` $:= \exp(2\pi\sqrt{-1}/(q^i-1))$, i a positive integer,

`GEWYi` $:= \exp(2\pi\sqrt{-1}/(q^i+1))$, i a positive integer,

`GEWCi` $:= \exp(2\pi\sqrt{-1}/\Phi_i(q))$, i a positive integer, $\Phi_i :=$ i-th cyclotomic polynomial.

Various combinations and variants of these are used according to the following table.

Name	Which Root	Pretty Printed Name
`GEWZ1`	$q-1$	ζ_1
`GEWY1`	$q+1$	ξ_1
`GEW2Z1`	$2(q-1)$	ρ_1
`GEWZ2`	q^2-1	ζ_2
`GEWY2`	q^2+1	ξ_2
`GEWC3`	q^2+q+1	φ_3
`GEWC6`	q^2-q+1	φ_6
`GEWC12`	q^4-q^2+1	φ_{12}
`GEWC8p`	$q^2+\sqrt{2}q+1$	${\varphi_8}'$
`GEWC8pp`	$q^2-\sqrt{2}q+1$	${\varphi_8}''$
`GEWC8pZ2`	$(q^2+\sqrt{2}q+1)(q^2-1)$	${\psi_8}'$
`GEWC8ppZ2`	$(q^2-\sqrt{2}q+1)(q^2-1)$	${\psi_8}''$
`GEWC12p`	$q^2-\sqrt{3}q+1$	${\varphi_{12}}'$
`GEWC12pp`	$q^2+\sqrt{3}q+1$	${\varphi_{12}}''$
`GEWC24p`	$q^4+\sqrt{2}q^3+q^2+\sqrt{2}q+1$	${\varphi_{24}}'$
`GEWC24pp`	$q^4-\sqrt{2}q^3+q^2-\sqrt{2}q+1$	${\varphi_{24}}''$
`GEWZ1Y2`	$(q-1)(q^2+1)$	η_2
`GEWZ3`	q^3-1	ζ_3
`GEWY3`	q^3+1	ξ_3
`GEWZ1Y3`	$(q-1)(q^3+1)$	η_3
`GEWY3Z1`	$(q-1)(q^3+1)$	η_3
`GEWY1Z3`	$(q+1)(q^3-1)$	μ_3
`GEWZ3Y1`	$(q+1)(q^3-1)$	μ_3
`GEWZ4`	q^4-1	ζ_4
`GEWY4`	q^4+1	ξ_4

Chapter 3

Generic Character Tables and Green Functions

We give a detailed description of the commands of the MAPLE part of CHEVIE for working with generic character tables. In the following, $\mathbb{G}$ denotes a generic finite reductive group. For the formal parameters of the functions we use an italic font. The most common parameters and their data types are the following.

Parameter	Data Type
t	Generic character table or table of Green functions (a MAPLE array)
l, $l1$, $l2$, ...	List of integers (a MAPLE list)
i, $i1$, $i2$, ...	Integer (a MAPLE integer)

All functions which allow lists of integers l, $l1$, ... as parameters can also be called with integers i, $i1$, ... instead of the lists. Some functions (e.g. `NrChars`, `Norm`) have a different behaviour when called with integer parameters—they return a value rather than just printing it onto the screen. This is always explained in the description of the functions. In all other cases, an integer parameter is only a short-cut for a list with one entry. This is not always mentioned explicitly.

3.1 GenCharTab

`GenCharTab(` ‘ f ‘ `)`

Reads the generic character table stored on the file f. Please note the back quotes in the function call. The file name f indicates the series of groups of Lie type of which the generic character table is stored on f. The MAPLE array containing the generic table also has the name f.

The MAPLE procedures needed to work with the generic table are also read.

After reading the generic table, MAPLE assigns the array containing the character table to the variable `g`. If you do not like this name, change it by typing: `newname` := f;

To find out which files are available, call the function with no parameter at all.

```
> GenCharTab(`SL2.0`);
**************************************************************************
*                                                                        *
*                                                                        *
*              Generic Character Table of SL_2(q), q even                *
*                                                                        *
*                                                                        *
**************************************************************************
                                g := `SL2.0`

> GenCharTab();
The following generic character tables are available:

GU3      PGU3.2   PGU3.n2  PSU3.n2  SU3.2    SU3.n2
2B2  Sz
2F4  Ree
2G2  ree
3D4.0  3D4.1
GL2     GU2     PGL2.0  PGL2.1  PSL2.0  PSL2.1  PSL2.3  SL2.0   SL2.1
GL3      PGL3.1   PGL3.n1  PSL3.n1  SL3.1    SL3.n1
Sp4.0      uniCSp4.1
CSp6.1     CSp6_1.m   Sp6.0      Sp6_0.m    uniCSp6.1  uniSp6.0
uniCSpin8.1  uniSpin8.0
G2.01  G2.02  G2.10  G2.11  G2.12
```

3.2 GreenFunTab

`GreenFunTab(` ` *f* ` `)`

Reads the table of Green functions stored on the file f. Please note the back quotes in the function call. The file name f indicates the series of groups of Lie type of which the Green functions are stored on f. The MAPLE array containing the Green functions has the name f`.green`.

After reading the table of Green functions, MAPLE assigns the array containing the functions to the variable g. If you do not like this name, change it by typing: `newname` := *f.green* ;

To find out which files are available, call the function with no parameter at all.

```
> GreenFunTab(`GL5`);
**************************************************************************
*                                                                        *
*                                                                        *
*                    Green Functions of GL_5(q)                          *
*                                                                        *
*                                                                        *
**************************************************************************
                               g := `GL5green`
```

3.3 CentOrd

CentOrd(t)
CentOrd(t , l)
CentOrd(t , i)

Prints the centralizer orders of the class types of t in factored form.

If the second parameter is a list l , then only the centralizer orders corresponding to the class types specified by l are printed. The function returns nothing.

If the second parameter is an integer i , then the centralizer order of class type i is returned.

```
> CentOrd(GL3);

clt   Order of centralizer

=======================================================
1   q^3*(q-1)^3*(q+1)*(q^2+q+1)
2   (q-1)^2*q^3
3   q^2*(q-1)
4   q*(q-1)^3*(q+1)
5   (q-1)^2*q
6   (q-1)^3
7   (q-1)^2*(q+1)
8   (q-1)*(q^2+q+1)
> CentOrd(GL3,[3..6]);

clt   Order of centralizer(s)

=======================================================
3   q^2*(q-1)
4   q*(q-1)^3*(q+1)
5   (q-1)^2*q
6   (q-1)^3
> c8 := CentOrd(GL3,8):
> c8;

                                      2
                         (q - 1) (q  + q + 1)
```

3.4 CharDeg

CharDeg(t)
CharDeg(t , l)
CharDeg(t , i)

Prints the degrees of the character types of t in factored form.

If the second parameter is a list l, then only the character degrees corresponding to the character types specified by l are printed. The function returns nothing.

If the second parameter is an integer i, then the character degree of character type i is returned.

```
 > CharDeg(`PSL2.1`);

cht   Degree of character(s)

=======================================================
1   1
2   q
3   1/2*q+1/2
4   1/2*q+1/2
5   q+1
6   q-1
> d4 := CharDeg(`PSL2.1`,4):
> d4;

                                  1/2 q + 1/2
```

3.5 NrChars

NrChars(t)
NrChars(t , l)
NrChars(t , i)

Prints the number of characters in the various character types of t.

If the parameter l is present, then only the numbers corresponding to the character types specified by l are printed.

If the second parameter is an integer i, then the number of characters of character type i is returned.

```
> NrChars(Sz);

cht   Number of characters in this type

=======================================================
1   1
2   1
3   1
4   1
5   1/2*q^2-1
6   1/4*q*(q+2^(1/2))
7   1/4*q*(q-2^(1/2))
> nr6 := NrChars(Sz,6):
```

```
> nr6;

                                          1/2
                              1/4 q (q + 2   )
> Status(Sz);

Status of   Sz

=======================================================
Order                     :   q^4*(q^4+1)*(q^2-1)
Number of character types  :   7
Number of rows available   :   7
Number of class types      :   7
Number of columns available:   7
> ord := 0:
> for cc from 1 to 7 do
>     ord := factor(ord + NrChars(Sz,cc) * CharDeg(Sz,cc)^2)
> od:
> ord;
               2    1/2           2    1/2                         4
             (q  + 2    q + 1) (q  - 2    q + 1) (q + 1) (q - 1) q
```

3.6 NrClasses

NrClasses(t)
NrClasses(t , l)
NrClasses(t , i)

Prints the number of conjugacy classes in the various class types of t .

If the parameter l is present, then only the numbers corresponding to the class types specified by l are printed.

If the second parameter is an integer i , then the number of conjugacy classes of class type i is returned.

```
> NrClasses(GL3);

clt   Number of classes in this type

=======================================================
1   q-1
2   q-1
3   q-1
4   (q-2)*(q-1)
5   (q-2)*(q-1)
6   1/6*(q-1)*(q-2)*(q-3)
7   1/2*q*(q-1)^2
8   1/3*q*(q-1)*(q+1)
> nr6 := NrClasses(GL3,6):
```

```
> nr6;

                    1/6 (q - 1) (q - 2) (q - 3)
```

3.7 PrintCharParam

`PrintCharParam(` t `)`
`PrintCharParam(` t `,` l `)`
`PrintCharParam(` t `,` i `)`

Prints the names and ranges of the parameters which distinguish the irreducible characters inside a character type.

If the parameter l or i is present, the information is given only for the character types specified by l or i .

The information on the parameters is given in form of a table with two or three columns, the first for the list of parameters, the second for the list of exceptions. If some of the parameters have been substituted with the function `SpecCharParam`, then the table of parameters contains a third column with the list of substitutions.

The list of parameters is of the form:

```
[ p_1 = 1 .. n_1, p_2 = 1 .. n_2, ... , p_r = 1 .. n_r ].
```

The p_j's are the names of the parameters. The expressions n_j are polynomials in q with coefficients in $\mathbb{Q}$ (or in $\mathbb{Q}[\frac{1}{\sqrt{2}}]$ for the Suzuki groups ${}^2B_2(q^2)$ or the Ree groups ${}^2F_4(q^2)$, respectively $\mathbb{Q}[\frac{1}{\sqrt{3}}]$ for the Ree groups ${}^2G_2(q^2)$).

The parameter vector

$$(p_1, p_2, \ldots, p_r)$$

can be substituted by an element of $\mathbb{Z}^r$, but two vectors which differ by an element of $n_1\mathbb{Z} \times n_2\mathbb{Z} \times \cdots \times n_r\mathbb{Z}$ yield the same character. In other words, the parameter vector can be taken to be an element of $\mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_r\mathbb{Z}$. This group is called the *character parameter group* in the following.

The list of exceptions is of the form:

```
[ [ expr_1, m_1 ], [ expr_2, m_2 ], ... , [ expr_s, m_s] ].
```

Here, `expr_j` is an expression in the parameters, and `m_j` a polynomial in q with coefficients in $\mathbb{Q}$ (or in $\mathbb{Q}[\frac{1}{\sqrt{2}}]$ for the Suzuki groups ${}^2B_2(q^2)$ or the Ree groups ${}^2F_4(q^2)$ respectively $\mathbb{Q}[\frac{1}{\sqrt{3}}]$ for the Ree groups ${}^2G_2(q^2)$). Those parameters have to be excluded, for which `m_j` divides `expr_j`. The remaining parameters are called *admissible* in the following.

Warning: Different values of admissible parameters may give the same characters.

The list of substitutions is of the form:

`[ { set_1 }, { set_2 }, ... ].`

`{ set_j }` contains the set of substitutions from the `j`-th call of the function `SpecCharParam`.

```
> PrintCharParam(GL3,[5,7]);

cht   Parameters   Exceptions

========================================================
5   [nN = 1 .. q-1, mM = 1 .. q-1]   [[nN-mM, q-1]]
7   [mM = 1 .. q-1, nN = 1 .. q^2-1]   [[nN, q+1]]
```

The characters of type 5 in $GL_3(q)$ are parametrized by elements $(nN, mM) \in \mathbb{Z}/(q-1)\mathbb{Z} \times \mathbb{Z}/(q-1)\mathbb{Z}$, $nN \not\equiv mM (\text{mod } q-1)$. The characters of type 7 in $GL_3(q)$ are parametrized by elements $(mM, nN) \in \mathbb{Z}/(q-1)\mathbb{Z} \times \mathbb{Z}/(q^2-1)\mathbb{Z}$, $(q+1) \not| nN$.

```
> SpecCharParam(GL3,5,mM = - nN);
Substituted
      {mM = -nN}
in character type 5.
> PrintCharParam(GL3,5);

cht   Parameters   Exceptions   Substitutions

========================================================
5   [nN = 1 .. q-1, mM = 1 .. q-1]   [[nN-mM, q-1]]   [{mM = -nN}]
> SpecCharParam(GL3,5,nN = 1);
Substituted
      {nN = 1}
in character type 5.
> PrintCharParam(GL3,5);

cht   Parameters   Exceptions   Substitutions

========================================================
5   [nN = 1 .. q-1, mM = 1 .. q-1]   [[nN-mM, q-1]]   [{mM = -nN}, {nN = 1}]
```

3.8 PrintClassParam

`PrintClassParam(` t `)`
`PrintClassParam(` t `,` l `)`
`PrintClassParam(` t `,` i `)`

Prints the names and ranges of the parameters which distinguish the conjugacy classes inside a class type.

If one of the parameters l or i is present, the information is given only for the class types specified by l or i.

The information on the parameters is given in form of a table with two or three columns, the first for the list of parameters, the second for the list of exceptions. If some of the parameters have been substituted with the function `SpecClassParam`, then the table of parameters contains a third column with the list of substitutions.

The list of parameters is of the form:

```
[ p_1 = 1 .. n_1, p_2 = 1 .. n_2, ... , p_r = 1 .. n_r ].
```

The p_j's are the names of the parameters. The expressions n_j are polynomials in q with coefficients in $\mathbb{Q}$ (or in $\mathbb{Q}[\frac{1}{\sqrt{2}}]$ for the Suzuki groups ${}^2B_2(q^2)$ or the Ree groups ${}^2F_4(q^2)$ respectively $\mathbb{Q}[\frac{1}{\sqrt{3}}]$ for the Ree groups ${}^2G_2(q^2)$).

The parameter vector

$$(p_1, p_2, \ldots, p_r)$$

can be substituted by an element of $\mathbb{Z}^r$, but two vectors which differ by an element of $n_1\mathbb{Z} \times n_2\mathbb{Z} \times \cdots \times n_r\mathbb{Z}$ yield the same character value. In other words, the parameter vector can be taken to be an element of $\mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_r\mathbb{Z}$. This group is called the *class parameter group* in the following.

The list of exceptions is of the form

```
[ [ expr_1, m_1 ], [ expr_2, m_2 ], ... , [ expr_s, m_s] ].
```

Here, `expr_j` is an expression in the parameters, and `m_j` a polynomial in q with coefficients in $\mathbb{Q}$ (or in $\mathbb{Q}[\frac{1}{\sqrt{2}}]$ for the Suzuki groups ${}^2B_2(q^2)$ or the Ree groups ${}^2F_4(q^2)$ respectively $\mathbb{Q}[\frac{1}{\sqrt{3}}]$ for the Ree groups ${}^2G_2(q^2)$). Those parameters have to be excluded, for which `m_j` divides `expr_j`. The remaining parameters are called *admissible* in the following.

Warning: Different values of admissible parameters may give the same character values.

The list of substitutions is of the form:

```
[ { set_1 }, { set_2 }, ... ].
```

`{ set_j }` contains the set of substitutions from the `j`-th call of the function `SpecClassParam`.

```
> PrintClassParam(GU3,[4,8]);

clt   Parameters   Exceptions

=====================================================
4   [kK = 1 .. q+1, lL = 1 .. q+1]   [[kK-lL, q+1]]
8   [kK = 1 .. q^3+1]   [[kK, q^2-q+1]]
```

See the example for `PrintCharParam` for a detailed explanation.

```
> SpecClassParam(GU3,4,lL=2*kK);
Substituted
      {lL = 2*kK}
in class type 4.
> SpecClassParam(GU3,4,kK=1);
Substituted
      {kK = 1}
in class type 4.
> PrintClassParam(GU3,4);

clt   Parameters   Exceptions   Substitutions

========================================================
4   [kK = 1 .. q+1, lL = 1 .. q+1]   [[kK-lL, q+1]]   [{lL = 2*kK}, {kK = 1}]
```

3.9 PrintInfoChar

`PrintInfoChar(` t `)`
`PrintInfoChar(` t `,` l `)`

Prints information about the character types of the generic character table t respectively the Green functions of the table of Green functions t.

If the parameter l is present, only the information for the character types specified by l is printed.

The information corresponding to character type i of table t is stored in position `[i,-1]` of the MAPLE array containing t. The information is given in form of a list, called information list in the following.

If t is a table of Green functions, the information list contains exactly one entry. This is a symbol for the F-conjugacy class of the Weyl group of G which corresponds to the `i`-th Green function on t.

The following description is relevant only in case t is a generic character table. Here, the information list contains a varying number of entries.

Position 1 of the information list contains miscellaneous information ranging from the empty string (for no information at all) to the names of the tensor factors, if the `i`-th character type of table t happens to be a tensor product.

Position 2 of the information list contains a list `[a, b]`. The integer `a` numbers the semisimple character types of t, beginning with 1 for the character type containing the trivial character. The integer `b` numbers the unipotent characters of the centralizer of the semisimple class type corresponding to a in the dual group, beginning with 0 for the trivial character. Thus `[1, b]` denotes the `(b-1)`-st unipotent character, and `[a, 0]` the `a`-th semisimple character type of the group.

Position 3 of the information list contains a list `[s, p]` which corresponds to `[a, b]`. The string `s` indicates the Dynkin diagram, including automorphism, of the semisimple part of the centralizer of the semisimple class type corresponding to `a` in the dual group. If this centralizer is a torus, we write `A_0` for `s`. The entry `p` informs about the `(b-1)`-st unipotent character of this centralizer. The data type of `p` depends on the group. For classical groups, `p` will usually be a partition, a pair of partitions, a multi-partition or a symbol. A partition is given as a list of its parts in non-increasing order. This description is, strictly speaking, only valid, if t is the generic character table of a finite reductive group arising from an algebraic group with a connected centre such that the dual group also has connected centre. Otherwise, it has to be adjusted suitably. The preceding remark only concerns the tables for $PSL_2(q)$, $SL_2(q)$, $PGL_2(q)$, q odd, $SL_3(q)$, $PGL_3(q)$, for 3 dividing $q-1$, and $SU_3(q)$, $PGU_3(q)$, for 3 dividing $q+1$.

Other positions of the information list may contain additional information such as for example different notations for the characters. To find out about additional information use the function `PrintInfoTab`.

```
> PrintInfoChar(`G2.10`);

cht   Information

=========================================================
1    [ , [1, 0], [G_2, \phi_{1,0}], X_{11}, \vartheta_0]
2    [ , [1, 1], [G_2, \phi_{2,1}], X_{16}, \vartheta_1]
3    [ , [1, 2], [G_2, \phi_{2,2}], X_{15}, \vartheta_2]
4    [ , [1, 3], [G_2, \phi_{1,3}'], X_{13}, \vartheta_3]
5    [ , [1, 4], [G_2, \phi_{1,3}''], X_{14}, \vartheta_4]
6    [ , [1, 5], [G_2, G_2[1]], X_{18}, \vartheta_{10}]
7    [ , [1, 6], [G_2, G_2[-1]], X_{17}, \vartheta_{11}]
8    [ , [1, 7], [G_2, G_2[\theta]], X_{19}(1), \vartheta_{12}(1)]
9    [ , [1, 8], [G_2, G_2[\theta^2]], X_{19}(2), \vartheta_{12}(-1)]
10    [ , [1, 9], [G_2, \phi_{1,6}], X_{12}, \vartheta_5]
11    [ , [2, 0], [A_1 + A_1, [[2], [2]]], X_{22}, \vartheta_6]
12    [ , [2, 1], [A_1 + A_1, [[2], [1, 1]]], X_{23}, \vartheta_8]
13    [ , [2, 2], [A_1 + A_1, [[1, 1], [2]]], X_{24}, \vartheta_9]
14    [ , [2, 3], [A_1 + A_1, [[1, 1], [1, 1]]], X_{21}, \vartheta_7]
15    [ , [5, 0], [A_1, [2]], X_{1a}', \chi_1(k)]
16    [ , [5, 1], [A_1, [1, 1]], X_{1a}, \chi_2(k)]
17    [ , [4, 0], [A_1, [2]], X_{1b}', \chi_3(k)]
18    [ , [4, 1], [A_1, [1, 1]], X_{1b}, \chi_4(k)]
19    [ , [7, 0], [A_1, [2]], X_{2a}', \chi_7(k)]
20    [ , [7, 1], [A_1, [1, 1]], X_{2a}, \chi_8(k)]
21    [ , [6, 0], [A_1, [2]], X_{2b}', \chi_5(k)]
22    [ , [6, 1], [A_1, [1, 1]], X_{2b}, \chi_6(k)]
23    [ , [8, 0], [A_0, [1]], X_{1}, \chi_9(k,l)]
24    [ , [9, 0], [A_0, [1]], X_{2}, \chi_{12}(k,l)]
25    [ , [11, 0], [A_0, [1]], X_{a}, \chi_{10}(k)]
26    [ , [10, 0], [A_0, [1]], X_{b}, \chi_{11}(k)]
```

```
27   [ , [12, 0], [A_0, [1]], X_{3}, \chi_{13}(k)]
28   [ , [13, 0], [A_0, [1]], X_{6}, \chi_{14}(k)]
```

3.10 PrintInfoClass

`PrintInfoClass(` t `)`
`PrintInfoClass(` t `,` l `)`

Prints information about the class types of table t.

If the parameter l is present, only the information for the class types specified by l is printed.

The information corresponding to class type i of table t is stored in position `[-1,i]` of the MAPLE array containing t. It is given in form of a list, called information list in the following.

If t is a table of Green functions, the information list contains exactly one entry: A symbol for the `i`-th unipotent conjugacy class.

If t is a generic character table, the information list contains a varying number of entries.

Position 1 of the information list contains miscellaneous information.

Position 2 of the information list contains a list `[a, b]`. The integer a numbers the semisimple section types of t, beginning with 1 for the type containing the unit element. The integer `b` numbers the unipotent conjugacy classes of the centralizer of the semisimple section type `a`, beginning with 0 for the unipotent class containing the unit element. Thus `[1, b]` denotes the `(b-1)`-st unipotent conjugacy class, and `[a, 0]` the `a`-th semisimple class type of the group.

Position 3 of the information list contains a list `[s, p]` which corresponds to `[a, b]`. The string `s` indicates the Dynkin diagram, including automorphism, of the semisimple part of the centralizer of the semisimple section type `a`. If this centralizer is a torus, we write `A_0` for `s`. The entry `p` informs about the `(b-1)`-st unipotent conjugacy class of this centralizer. The data type of `p` depends on the group. For classical groups `p` will be a partition or a pair of partitions. A partition is given as a list of its parts in non-increasing order.

Other positions of the information list may contain additional information such as for example different notations for the conjugacy classes. To find out about additional information use the function `PrintInfoTab`.

```
> PrintInfoClass(GL3);

clt   Information

=======================================================
1   [ , [1, 0], [A_2, [1, 1, 1]]]
2   [ , [1, 1], [A_2, [2, 1]]]
3   [ , [1, 2], [A_2, [3]]]
4   [ , [2, 0], [A_1, [1, 1]]]
5   [ , [2, 1], [A_1, [2]]]
```

```
6   [ , [3, 0], [A_0, [1]]]
7   [ , [4, 0], [A_0, [1]]]
8   [ , [5, 0], [A_0, [1]]]
```

3.11 PrintInfoTab

`PrintInfoTab(` *t* `)`

Prints useful information about the generic character table or table of Green functions t (partly) in TEX-format. It gives a reference to the author(s) of the table and possibly some additional relevant information.

```
- Information about the generic character table of $G_2(q)$,
   $q$ even, congruent to $2$ modulo $3$

- CHEVIE-name of the table: `G2.02`

- The table was first computed in:
  {\sc H.~Enomoto and H.~Yamada}, The characters of $G_2(2^n)$,
  {\em Japan.\ J.~Math.}, {\bf 12} (1986), 325--377.

- See also:
  {\sc B.~Chang and R.~Ree}, The characters of $G_2(q)$, {\em in}:
  Symposia Mathematica Vol.~13, pp.~395--413, Academic Press, London, 1974.

  {\sc G.~Hiss}, Zerlegungszahlen endlicher Gruppen vom Lie-Typ in
  nicht-definierender Charakteristik, Habilitationsschrift, Aachen 1990.

- Note:
  Enomoto's and Yamada's notation for the irreducible characters is
  given in the fifth component of the character information list.

  An equivalent to Chang's and Ree's notation for the irreducible
  characters is given in the fourth component of the character
  information list.

  Enomoto's and Yamada's notation for the conjugacy classes is given in
  the fourth component of the class information list.

- Example:
> PrintInfoChar(`G2.02`,6);

cht   Information

=====================================================
6   [ , [1, 5], [G_2, G_2[1]], X_{18}, \vartheta_1']
```

```
- Explanation of example:
  Character type 6 of G2.02 is called \vartheta_1' by Enomoto--Yamada
  and corresponds to X_{18} in Chang--Ree.
```

3.12 PrintToTeX

`PrintToTeX(` *t* `)`
`PrintToTeX(` *t* , *l1* `)`
`PrintToTeX(` *t* , *l1* , *l2* `)`
`PrintToTeXPhi(` *t* `)`
`PrintToTeXPhi(` *t* , *l1* `)`
`PrintToTeXPhi(` *t* , *l1* , *l2* `)`

Prints the values of the characters of the generic character table or table of Green functions t onto a TEX file `tempptt.tex` and runs LATEX on `tempptt.tex`. The function can take an additional parameter s , a MAPLE string, in which case the TEX file is given the name `s.tex`.

If a non-empty string `preview_prog` is assigned to the global MAPLE variable `PREVIEWNAME`, then the TEX-preview program `preview_prog` is called with the file `tempptt.dvi`, respectively `s.dvi`. The default value of `PREVIEWNAME` is `xdvi`. It can be changed upon installing CHEVIE (cf. the `README` file in `chevie/doc`).

If the parameter *l1* is present, only the values of the character types specified by *l1* are printed.

The additional parameter *l2* specifies the class types whose values are to be printed.

With the command `PrintToTeX` the values are printed similar to their format in the table.

The command PrintToTeXPhi can be used for tables with polynomial entries (tables of Green functions or unipotent characters). Here the character values are first factorized and then each factor which is the i-th cyclotomic polynomial in q is substituted by `phi.i` and printed as ϕ_i.

3.13 PrintVal

`PrintVal(` *t* `)`
`PrintVal(` *t* , *l1* `)`
`PrintVal(` *t* , *l1* , *l2* `)`
`PrintValPhi (` *t* `)`
`PrintValPhi (` *t* , *l1* `)`
`PrintValPhi (` *t* , *l1* , *l2* `)`

Prints the values of the characters or Green functions of table t .

If the parameter *l1* is present, only the values of the character types specified by *l1* are printed.

The additional parameter *l2* specifies the class types whose values are to be printed.

The form `PrintValPhi` of this command can be used for tables with polynomial entries, for example tables of Green functions or unipotent characters. It factors the character values and substitutes the i-th cyclotomic polynomial in q into `phi.i`. Entries which are zero are marked by a dot.

These values are only printed to the screen and not returned to the system. The values are stored on the MAPLE array t containing the table; so `v := t[ a, b ]` assigns the value of character type `a` on class type `b` to the variable `v`.

```
> PrintVal(`SL2.0`);

clt   Value of character type    1    on class type clt

1   1
2   1
3   1
4   1

clt   Value of character type    2    on class type clt

1   q
2   0
3   1
4   -1

clt   Value of character type    3    on class type clt

1   q+1
2   1
3   GEWZ1^(nN*aA)+GEWZ1^(-nN*aA)
4   0

clt   Value of character type    4    on class type clt

1   q-1
2   -1
3   0
4   -GEWY1^(nN*aA)-GEWY1^(-nN*aA)
> v := g[4,1]:
> v;

                                   q - 1
```

3.14 Status

`Status(` t `)`

Prints information about the generic character table or table of Green functions t .

The following information is printed:

- The "Order" of the group.
- The "Number of character types" stored on t .
- The "Number of rows available" on the MAPLE array containing t .
- The "Number of class types" stored on t .
- The "Number of columns available" on the MAPLE array containing t .

```
> Status(GL3);

Status of   GL3

=======================================================
Order                      :   q^3*(q-1)^3*(q+1)*(q^2+q+1)
Number of character types  :   8
Number of rows available   :   8
Number of class types      :   8
Number of columns available:   8
```

3.15 Copy

`Copy(` ta , tn , $l1$, $l2$, `'ROWS'+` n , `'COLS'+` m `)`

Creates a copy tn of table ta .

The last four parameters are optional. They may appear in any number and order, but please note the following conventions.

The two parameters $l1$, $l2$ specify the character types respectively class types to be copied. If just one of these parameters is present, it will refer to the character types. The default is to copy all characters on all classes.

The remaining two parameters are used to increase the number of rows respectively the number of columns of the table tn. If `'ROWS'` + n is present, tn will have n more rows available than ta . Similarly, the number of columns available for tn is increased by m , if the parameter `'COLS'` + m is given. The integers m and n may be negative, in which case the number of columns respective rows is decreased.

Please note the quotes in `'ROWS'` + n , `'COLS'` + m .

```
> Copy(GU3,new,[],[],'ROWS' + 5);
> Status(new);

Status of   new

=======================================================
Order                      :   q^3*(q+1)^3*(q-1)*(q^2-q+1)
Number of character types  :   0
```

```
Number of rows available   :   13
Number of class types      :   0
Number of columns available:   8
> Copy(GU3,new1,1,[1..4],'COLS' + 2);
> Status(new1);

Status of   new1

=======================================================
Order                      :   q^3*(q+1)^3*(q-1)*(q^2-q+1)
Number of character types  :   1
Number of rows available   :   8
Number of class types      :   4
Number of columns available:   10
> PrintVal(new1);

clt   Value of character type    1    on class type clt

1   GEWY1^(3*uU*kK)
2   GEWY1^(3*uU*kK)
3   GEWY1^(3*uU*kK)
4   GEWY1^(2*uU*kK+1L*uU)
```

3.16 CopyChar

CopyChar(*t1* , *t2*)
CopyChar(*t1* , *t2* , *l*)

Appends character types or Green functions of table *t1* to table *t2* . This table must be present and must have enough rows.

The third parameter, if present, specifies the character type(s) to be copied. The default is to append all character types.

```
> Copy(GU3,new,[],[]);
> CopyChar(GU3,new,[2,3,8]);
> Status(new);

Status of   new

=======================================================
Order                      :   q^3*(q+1)^3*(q-1)*(q^2-q+1)
Number of character types  :   3
Number of rows available   :   8
Number of class types      :   8
Number of columns available:   8
> CopyChar(GU3,new);
Error, (in CopyChar) Second table too small: Enlarge with "Copy"
```

3.17 CopyClass

`CopyClass(` *t1* `,` *t2* `)`
`CopyClass(` *t1* `,` *t2* `,` *l* `)`

Appends class types (columns) of table *t1* to table *t2*. This table must be present and must have enough columns.

The third parameter, if present, specifies the class type to be copied. The default is to append all class types.

```
> Copy(GU3,new,[],[]);
> CopyClass(GU3,new,8);
> Status(new);

Status of   new

=======================================================
Order                      :   0
Number of character types  :   8
Number of rows available   :   8
Number of class types      :   1
Number of columns available:   8
```

3.18 LinComb

`LinComb(` *t1* `,` *c1* `,` *i1* `, ... ,` *tn* `)`

Computes linear combinations of character types.

The function has $3k+1$ arguments. They must be periodically a generic character table *tj*, a coefficient *cj* and a number *ij* $(j = 1, \ldots, k)$. The function then computes the sum over the *cj* times the *ij*-th character type of table *tj*. The result is appended to the table *tn* (the last argument).

The character parameters in the resulting character type are obtained from the original parameters by appending *lj* to the parameters of character type *ij* in table *tj* (for $j = 1, \ldots, k$).

You can find out about the characters in a linear combination stored on table *tn* with the function `PrintInfoChar`.

```
> GenCharTab(GL3);
************************************************************************
*                                                                      *
*                                                                      *
*                   Generic Character Table of GL_3(q)                 *
*                                                                      *
*                                                                      *
************************************************************************
```

```
                                 g := `GL3`

> Copy(GL3,new,[],[]);
> LinComb(GL3,1,1,GL3,1,2,GL3,1,3,GL3,1,4,new);
> # all sums of four characters with one character from each character type
> PrintInfoChar(new,1);

cht   Information

=======================================================
1   [LinComb(GL3,1,1,GL3,1,2,GL3,1,3,GL3,1,4)]
> PrintCharParam(new,1);

cht   Parameters   Exceptions

========================================================
1   [nNl1 = 1 .. q-1, nNl2 = 1 .. q-1, nNl3 = 1 .. q-1, nNl4 = 1 .. q-1, mMl4
 = 1 .. q-1]   [[nNl4-mMl4, q-1]]

SEE ALSO:
Copy, CopyChar, SpecCharParam, PrintInfoChar, CHEVIE (for an overview)
```

3.19 ClassMult

`ClassMult(` *t* `)`
`ClassMult(` *t* , *l1* , *l2* , *l3* `)`
`ClassMult(` *t* , *l1* , *l2* `)`
`ClassMult(` *t* , *l1* `)`
`ClassMult(` *t* , *i1* , *i2* , *i3* `)`

Calculates the (generic) class multiplication constants of the class types of table *t* (structure constants).

The additional parameters of the function, if present, specify the class types for which the multiplication constants are calculated. Omitted lists are interpreted as full lists.

A message `Possible Exceptions` informs you about those values of the class parameters, for which the calculation is not valid (cf. the section about `Exceptions`). In the message, the integer `i, i = 1, 2, 3`, is appended to the class parameters of the factor `i` going into the computation of the multiplication constant.

Similar remarks as for `Norm` and `Scalar` apply.

The first four forms of the function only print the results but return nothing. If the last three parameters are integers *i1* , *i2* , *i3* , then a list with two entries is returned. The first entry is the (generic) class multiplication constant of the class types *i1* , *i2* , *i3* , the second entry the set of possible exceptions.

```
>  ClassMult(`SL2.0`,2,2);
ClassMult_SL2.0(2,2,1)=
                                     2
```

```
                                    q  - 1


ClassMult_SL2.0(2,2,2)=
                                    q - 2


Possible Exceptions:     {[aA3, q-1]}
ClassMult_SL2.0(2,2,3)=
                                    q - 1


Possible Exceptions:     {[aA3, q+1]}
ClassMult_SL2.0(2,2,4)=
                                    q + 1


> c244 := ClassMult(`SL2.0`,2,4,4):
> c244;

               [q + 1, {[aA2 - aA3, q + 1], [aA2 + aA3, q + 1]}]
```

3.20 Norm

`Norm(` *t* `)`
`Norm(` *t* , *l* `)`
`Norm(` *t* , *i* `)`

Calculates the (generic) norm of the character types of table *t* .

If the second parameter is a list *l* , only the norms of the character types specified by *l* are calculated.

A message `Possible Exceptions` informs you about those values of the parameters, for which the calculation is not valid (cf. the section about `Exceptions`).

The first two forms of the function only print the results but return nothing. If the second parameter is an integer *i* , then a list with two entries is returned. The first entry is the (generic) norm of the character type *i* , the second entry the set of possible exceptions.

```
> GenCharTab(`SL2.0`);
*************************************************************************
*                                                                       *
*                                                                       *
*               Generic Character Table of SL_2(q), q even              *
*                                                                       *
*                                                                       *
*************************************************************************
```

```
                              g := `SL2.0`

> Copy(`SL2.0`,h,[],[]);
> Tensor(`SL2.0`,4,`SL2.0`,4,h);
> PrintVal(h,1);

clt   Value of character type    1    on class type clt

1   (q-1)^2
2   1
3   0
4   GEWY1^(nNt1*aA+nNt2*aA)+GEWY1^(nNt1*aA-nNt2*aA)+GEWY1^(-nNt1*aA+nNt2*aA)+
GEWY1^(-nNt1*aA-nNt2*aA)
> Norm(h);
Possible Exceptions:     {[2*nNt1+2*nNt2, q+1], [2*nNt1-2*nNt2, q+1], [2*nNt1,
q+1], [2*nNt2, q+1]}
Norm_h(1)=
                                  q - 2


> n := Norm(h,1):
> n;

     [q - 2, {[2 nNt1 + 2 nNt2, q + 1], [2 nNt1, q + 1], [2 nNt2, q + 1],

                [2 nNt1 - 2 nNt2, q + 1]}                                 ]
```

In the above example, the characters of type 1 of table `h` are distinguished by the parameters `nNt1` and `nNt2` (cf. the section `Tensor`). The generic norm of such a character equals `q-2`, except if `q + 1` divides one of `2*nNt2-2*nNt1`, `2*nNt2`, `2*nNt2+2*nNt1` or `2*nNt1`. But `q` is even and each of `nNt1` and `nNt2` runs between 1 and `q`, so that the second and the last possibilities do not occur. Also, `q + 1` divides `2*nNt2-2*nNt1`, if and only if `q + 1` divides `nNt2-nNt1`. We may specialize the pair of parameters `(nNt1,nNt2)` and re-calculate the norm.

```
> PrintCharParam(h);

cht   Parameters   Exceptions

=======================================================
1   [nNt1 = 1 .. q+1, nNt2 = 1 .. q+1]   [[nNt1, q+1], [nNt2, q+1]]

> SpecCharParam(h,1,nNt2 = nNt1 + (q + 1)*fF);
Substituted
      {nNt2 = nNt1+(q+1)*fF}
```

```
in character type 1.
> PrintVal(h);

clt   Value of character type    1    on class type clt

1   (q-1)^2
2   1
3   0
4   2+GEWY1^(2*nNt1*aA)+GEWY1^(-2*nNt1*aA)
> Norm(h,1);

                            [q - 1, {[4 nNt1, q + 1]}]
```

3.21 Ortho2Norm

`Ortho2Norm(` *t* `)`
`Ortho2Norm(` *t* , *l* `)`
`Ortho2Norm(` *t* , *i* `)`

Calculates the (generic) norms of the class types of table *t* .

If the second parameter is a list *l* , only the norms of the class types specified by *l* are calculated.

A message `Possible Exceptions` informs you about those values of the parameters, for which the calculation is not valid (cf. the section about `Exceptions`).

The first two forms of the function only print the results but return nothing. If the second parameter is an integer *i* , then a list with two entries is returned. The first entry is the (generic) norm of the class type *i* , the second entry the set of possible exceptions.

```
> Ortho2Norm(`SL2.0`);
Ortho2Norm_SL2.0(1)=
                                        1


Ortho2Norm_SL2.0(2)=
                                        1


Possible Exceptions:     {[2*aA, q-1]}
Ortho2Norm_SL2.0(3)=
                                        1


Possible Exceptions:     {[2*aA, q+1]}
Ortho2Norm_SL2.0(4)=
                                        1
```

3.22 Ortho2Scalar

`Ortho2Scalar(` *t1* , *t2* `)`
`Ortho2Scalar(` *t1* `)`
`Ortho2Scalar(` *t1* , *l1* , *t2* , *l2* `)`
`Ortho2Scalar(` *t1* , *l1* , *t2* `)`
`Ortho2Scalar(` *t1* , *l1* `)`

Calculates the (generic) scalar products of the class types of table *t1* with those of table *t2* (column scalar products). If no table *t2* is specified, then *t2* is set to *t1* .

If the function is called with the single parameter *t1* , then only those scalar products are computed, for which the position of the second factor on the generic character table *t1* does not exceed the position of the first factor.

The additional parameters, if present, specify the character types, for which the scalar products are calculated.

A message **`Possible Exceptions`** informs you about those values of the parameters, for which the calculation is not valid (cf. the section about **`Exceptions`**). In the message, the integer `i, i = 1, 2`, is appended to the class parameters of the `i`-factor of the scalar product.

Usually the function only prints the results to the screen but returns nothing. If the second and fourth parameters are integers *l1, l2* , then a list with two entries is returned. The first entry is the (generic) scalar product of the class types *l1, l2* , the second entry the set of possible exceptions.

```
> o3 := Ortho2Scalar(`SL2.0`,3,`SL2.0`,3):
> o3;

                [0, {[- aA1 + aA2, q - 1], [aA1 + aA2, q - 1]}]

> PrintClassParam(`SL2.0`,3);

clt   Parameters   Exceptions

=====================================================
3   [aA = 1 .. q-1]   [[aA, q-1]]
```

3.23 Scalar

`Scalar(` *t1* , *t2* `)`
`Scalar(` *t1* `)`
`Scalar(` *t1* , *l1* , *t2* , *l2* `)`
`Scalar(` *t1* , *l1* , *t2* `)`
`Scalar(` *t1* , *l1* `)`

Calculates the (generic) scalar products of the character types of table *t1* with those of table *t2* (row scalar products). If no table *t2* is specified, then *t2* is set to *t1* .

If the function is called with the single parameter *t1* , then only those scalar products are computed, for which the position of the second factor on the generic character table *t1* does not exceed the position of the first factor.

The additional parameters, if present, specify the character types, for which the scalar products are calculated.

A message `Possible Exceptions` informs you about those values of the parameters, for which the calculation is not valid (cf. the section about `Exceptions`). In the message, the integer `i, i = 1, 2`, is appended to the character parameters of the `i`-factor of the scalar product. This does not apply for factors which have been constructed with the `Tensor` function (since the parameters of those have already been supplied with a suffix).

Usually the function only prints the results to the screen but returns nothing. If the second and fourth parameters are integers *l1, l2* , then a list with two entries is returned. The first entry is the (generic) scalar product of the character types *l1, l2* , the second entry the set of possible exceptions.

```
> GenCharTab(`SL2.0`);
*************************************************************************
*                                                                       *
*                                                                       *
*              Generic Character Table of SL_2(q), q even               *
*                                                                       *
*                                                                       *
*************************************************************************
                                 g := `SL2.0`

> Copy(`SL2.0`,h,[],[]);
> Tensor(`SL2.0`,4,`SL2.0`,4,h);
> PrintVal(h);

clt   Value of character type    1    on class type clt

1   (q-1)^2
2   1
3   0
4   GEWY1^(nNt1*aA+nNt2*aA)+GEWY1^(nNt1*aA-nNt2*aA)+GEWY1^(-nNt1*aA+nNt2*aA)+
GEWY1^(-nNt1*aA-nNt2*aA)
> SpecCharParam(h,1,nNt2 = nNt1);
Substituted
      {nNt2 = nNt1}
in character type 1.
> PrintVal(h);

clt   Value of character type    1    on class type clt

1   (q-1)^2
2   1
3   0
```

```
4   GEWY1^(2*nNt1*aA)+2+GEWY1^(-2*nNt1*aA)
> Norm(h);
Possible Exceptions:     {[4*nNt1, q+1]}
Norm_h(1)=
                                     q - 1



> Scalar(`SL2.0`,h);
Possible Exceptions:     {[2*nNt1, q+1]}
Scalar_SL2.0,h(1,1)=
                                       1


Possible Exceptions:     {[2*nNt1, q+1]}
Scalar_SL2.0,h(2,1)=
                                       0


Scalar_SL2.0,h(3,1)=
                                       1


Possible Exceptions:     {[2*nNt1-nN1, q+1], [2*nNt1+nN1, q+1], [nN1, q+1]}
Scalar_SL2.0,h(4,1)=
                                       1



> Copy(`SL2.0`,SL2new,[4,4]);
> PrintCharParam(SL2new);

cht   Parameters   Exceptions

=====================================================
1   [nN = 1 .. q+1]   [[nN, q+1]]
2   [nN = 1 .. q+1]   [[nN, q+1]]
> SpecCharParam(SL2new,1,nN = 2*nNt1 - (q + 1)*fF);
Substituted
      {nN = 2*nNt1-(q+1)*fF}
in character type 1.
> SpecCharParam(SL2new,2,nN = -2*nNt1 + (q + 1)*fF);
Substituted
      {nN = -2*nNt1+(q+1)*fF}
in character type 2.


> PrintVal(SL2new);
```

```
clt   Value of character type    1    on class type clt

1   q-1
2   -1
3   0
4   -GEWY1^(2*nNt1*aA)-GEWY1^(-2*nNt1*aA)

clt   Value of character type    2    on class type clt

1   q-1
2   -1
3   0
4   -GEWY1^(2*nNt1*aA)-GEWY1^(-2*nNt1*aA)
> Scalar(SL2new,1,h);
Possible Exceptions:     {[4*nNt1, q+1]}
Scalar_SL2new,h(1,1)=
                                        0
```

Substituting the two possible exceptional parameters for **nN** in character type 4 of `SL2.0` results in the same character, whose scalar product with the tensor product in question equals 0.

We thus have completely decomposed the tensor product $\chi_4(n) \otimes \chi_4(n)$. The result is

$$\chi_4(n) \otimes \chi_4(n) = \chi_1 + \sum_{m=1}^{(q-2)/2} \chi_3(m) + \sum_{m=1, m\neq 2n, q+1-2n}^{q/2} \chi_4(m).$$

Here we have written $\chi_i(n)$ for the character of type i with parameter n. For characters of type 3 we have $\chi_3(m) = \chi_3(q-1-m)$, $1 \leq m \leq (q-2)/2$, and so $\chi_3(m)$, $1 \leq m \leq (q-2)/2$ are all the distinct characters of type 3. Similarly, $\chi_4(m)$, $1 \leq m \leq q/2$, are the distinct characters of type 4.

3.24 SpecCharParam

`SpecCharParam(` *t* `,` *i* `,` *param_1 = expr_1* `,` *param_2 = expr_2* `, ... )`

Substitutes in character type *i* of table *t* the parameters *param_k* by the expressions *expr_k* , `k = 1, 2, ...`

```
> Copy(GL3,new,3);
> PrintVal(new);

clt   Value of character type    1    on class type clt

1   q^3*GEWZ1^(3*nN*aA)
2   0
```

```
3   0
4   q*GEWZ1^(2*nN*aA+nN*bB)
5   0
6   GEWZ1^(nN*aA+nN*bB+nN*cC)
7   -GEWZ1^(nN*aA+nN*bB)
8   GEWZ1^(nN*aA)
> PrintCharParam(new);

cht   Parameters   Exceptions

=======================================================
1   [nN = 1 .. q-1]   []
> SpecCharParam(new,1,nN=q-1);
Substituted
      {nN = q-1}
in character type 1.
> PrintVal(new);

clt   Value of character type    1    on class type clt

1   q^3
2   0
3   0
4   q
5   0
6   1
7   -1
8   1

> PrintCharParam(GL3,6);

cht   Parameters   Exceptions

=======================================================
6   [nN = 1 .. q-1, mM = 1 .. q-1, lL = 1 .. q-1]   [[nN-mM, q-1], [nN-lL, q-1]
, [mM-lL, q-1]]
> Copy(GL3,new,6);
> SpecCharParam(new,1,lL=nN,mM=nN);
Substituted
      {mM = nN, lL = nN}
in character type 1.
> Norm(GL3,[6]);
Possible Exceptions:     {[nN-mM, q-1], [nN-lL, q-1], [lL-mM, q-1]}
Norm_GL3(6)=
                                       1


> Norm(new,[1]);
```

```
Norm_new(1)=
                                        6
```

3.25 SpecClassParam

`SpecClassParam(` *t* , *i* , *param_1* = *expr_1* , *param_2* = *expr_2* , `...)`

Substitutes in class type *i* of table *t* the parameters *param_k* by the expressions *expr_k* , `k = 1, 2, ...`

```
> Copy(GL3,new,[1..8],3);
> PrintClassParam(new);

clt   Parameters   Exceptions

=======================================================
1   [aA = 1 .. q-1]    []
> SpecClassParam(new,1,aA=q-1);
Substituted
      {aA = q-1}
in class type 1.
```

3.26 Omega

`Omega(` *t1* , *t2* `)`
`Omega(` *t1* , *l* , *t2* `)`

Calculates the (generic) central characters corresponding to the character types of *t1* and writes them onto *t2* . Table *t2* must be present and have enough rows available.

If the second parameter *l* is a list or an integer, then only the central characters corresponding to the character types specified by *l* are computed.

```
> Copy(`SL2.0`,cen,[],[]);
> Omega(`SL2.0`,cen);
> PrintVal(cen);

clt   Value of character type    1    on class type clt

1   1
2   (q-1)*(q+1)
3   q*(q+1)
4   q*(q-1)

clt   Value of character type    2    on class type clt
```

```
1   1
2   0
3   q+1
4   -q+1

clt   Value of character type    3    on class type clt

1   1
2   q-1
3   (GEWZ1^(nN*aA)+GEWZ1^(-nN*aA))*q
4   0

clt   Value of character type    4    on class type clt

1   1
2   -q-1
3   0
4   -(GEWY1^(nN*aA)+GEWY1^(-nN*aA))*q
```

3.27 Tensor

`Tensor(` *t1* , *t2* , *t3* `)`
`Tensor(` *t1* , *t3* `)`
`Tensor(` *t1* , *l1* , *t2* , *l2* , *t3* `)`
`Tensor(` *t1* , *l1* , *t3* `)`

Computes the tensor products of the character types of table *t1* with the character types of table *t2* and writes them onto table *t3* . Table *t3* must be present and have enough rows available.

The tensor product of two character types is again a character type. Its parameters are obtained from the parameters of its `i`-th factors by appending `t.i`, `i = 1, 2`.

You can find out about the factors of the tensor products stored on table *t3* with the function `PrintInfoChar`.

The second form of the function does the same as the first for $t2 = t1$.

The parameters *l1* and *l2* may be integers or lists of integers. The parameter *l1*, if present, specifies the characters types of *t1* to be tensored with the character types of *t2*, specified by *l2*.

The fourth form of the function does the same as the third for $t2 = t1$ and takes *l2* to be the complete list.

```
> Copy(`SL2.0`,SL2new,[],[],'ROWS' + 10);
> Tensor(`SL2.0`,2,SL2new);
> Tensor(`SL2.0`,3,`SL2.0`,[3,4],SL2new);
> PrintInfoChar(SL2new);
```

```
cht   Information

=====================================================
1   [Tensor(SL2.0,2,SL2.0,1)]
2   [Tensor(SL2.0,2,SL2.0,2)]
3   [Tensor(SL2.0,2,SL2.0,3)]
4   [Tensor(SL2.0,2,SL2.0,4)]
5   [Tensor(SL2.0,3,SL2.0,3)]
6   [Tensor(SL2.0,3,SL2.0,4)]
```

3.28 GreenFunctionsA

`GreenFunctionsA(` i `)`
`GreenFunctionsA(` i `,` f `)`

Computes the Green functions for groups of type A_i. The result is returned as a list with two entries. The first entry of this list is the array of Green functions, whose rows and columns are parametrized by partitions of $i+1$. The second entry of the list contains the list of these partitions. (It works up to $i = 12$, i.e., GL_{13}.)

If the second parameter f is present, the result is written onto the file f in CHEVIE format.

```
> GreenFunctionsA(2);
Number of conjugacy classes:   3
Foulkes polynomials computed.
Now twisting to Green functions.
        [            2                   ]
        [  (q + 1) (q  + q + 1)   2 q + 1  1 ]
        [                                ]
        [             2                  ]
       [[ - (q - 1) (q  + q + 1)     1     1 ], [[1, 1, 1], [2, 1], [3]]]
        [                                ]
        [                   2            ]
        [    (q + 1) (q - 1)      - q + 1  1 ]
```

3.29 GreenFunctions2A

`GreenFunctions2A(` i `)`
`GreenFunctions2A(` i `,` f `)`

Computes the Green functions for groups of type ${}^2A_i$. The result is returned as a list with two entries. The first entry of this list is the array of Green functions, whose rows and columns are parametrized by partitions of $i+1$. The second entry of the list contains the list of these partitions. (It works up to $i = 12$, i.e., GU_{13}.)

If the second parameter f is present, the result is written onto the file f in CHEVIE format.

```
> GreenFunctions2A(2);
Number of conjugacy classes:   3
Foulkes polynomials computed.
Now twisting to Green functions.
       [             2                  ]
       [ - (q - 1) (q  - q + 1)  - 2 q + 1  1 ]
       [                                ]
       [            2                   ]
      [[  (q + 1) (q  - q + 1)       1      1 ], [[1, 1, 1], [2, 1], [3]]]
       [                                ]
       [                    2           ]
       [   - (q - 1) (q + 1)       q + 1    1 ]
```

Bibliography

[1] C. T. Benson and C. W. Curtis, *On the degrees and rationality of certain characters of finite Chevalley groups*, Trans. Amer. Math. Soc. **165** (1972), 251–273.

[2] M. Broué and G. Malle, *Théorèmes de Sylow génériques pour les groupes réductifs sur les corps finis*, Math. Ann. **292** (1992), 241–262.

[3] M. Broué, G. Malle, and J. Michel, *Generic blocks of finite reductive groups*, Représentations unipotentes génériques et blocs des groupes réductifs finis, Astérisque, vol. 212, Société Mathématique de France, 1993, pp. 7–92.

[4] R. W. Carter, *Finite groups of Lie type: Conjugacy classes and complex characters*, Wiley, New York, 1985.

[5] B. Chang and R. Ree, *The characters of* $G_2(q)$, Symposia Mathematica **XIII** (1974), 395–413.

[6] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt, *Maple V, Language Reference Manual*, Springer-Verlag, 1991.

[7] J. H. Conway, R. T. Curtis, S. P. Norton, R. A. Parker, and R. A. Wilson, *Atlas of Finite Groups*, Oxford University Press, London, 1984.

[8] P. Deligne and G. Lusztig, *Representations of reductive groups over finite fields*, Annals of Math. **103** (1976), 103–161.

[9] F. Digne and J. Michel, *Representations of finite groups of Lie type*, Cambridge University Press, Cambridge, 1991.

[10] M. Geck, *Eine Anwendung von MAPLE in der Darstellungstheorie der unitären Gruppen*, Diplomarbeit, Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 1988.

[11] ——, *Beiträge zur Darstellungstheorie von Iwahori–Hecke Algebren*, Habilitationsschrift, Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 1993.

[12] M. Geck and G. Pfeiffer, *On the irreducible characters of Hecke algebras*, Adv. in Math. **102** (1993), 79–94.

[13] J. A. Green, *The characters of the finite general linear groups*, Trans. Amer. Math. Soc. **80** (1955), 402–447.

[14] G. Hiss, *On the decomposition numbers of $G_2(q)$*, J. Algebra **120** (1989), 339–360.

[15] V. F. R. Jones, *Hecke algebra representations of braid groups and link polynomials*, Annals of Math. **126** (1987), 335–388.

[16] G. Lusztig, *Characters of reductive groups over a finite field*, Annals of Mathematical Studies, vol. 107, Princeton University Press, 1985.

[17] ______, *Remarks on computing irreducible characters*, J. Amer. Math. Soc. **5** (1992), 971–986.

[18] G. Malle, *Die unipotenten Charaktere von ${}^2F_4(q^2)$*, Comm. Algebra **18** (1990), 2361–2381.

[19] ______, *Generalized Deligne–Lusztig characters*, J. Algebra **159** (1993), 64–97.

[20] Martin Schönert et al., *GAP – Groups, Algorithms, and Programming*, Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, third ed., 1993.

[21] I. Schur, *Untersuchungen über die Darstellung der endlichen Gruppen durch gebrochene lineare Substitutionen*, J. reine angew. Math. **132** (1907), 85–137.

[22] B. Srinivasan, *The characters of the finite symplectic group $Sp(4,q)$*, Trans. Amer. Math. Soc. **131** (1968), 488–525.

Index