



```
> K:=cbasis(2):L:=gbasis(K$2);  
L := [Id &t Id, Id &t e1, Id &t e2, Id &t e12, e1 &t Id, e1 &t e1, e1 &t e2, e1 &t e12, e2 &t Id, e2 &t e1, e2 &t e2, e2 &t e12,  
      e12 &t Id, e12 &t e1, e12 &t e2, e12 &t e12]  
[ >  
[ >
```

**See Also:** [GTP:-`type/gradedmonom`](#), [GTP:-gbasis](#), [GTP:-`type/gradedodd`](#), [GTP:-grades](#), [Clifford:-`type/tensorprod`](#), [GTP:-gradedprods](#), [GTP:-gprods](#), [GTP:-`type/gradedeven`](#)

Last revised: November 5, 2002/RA

**Function:** GTP:-cmulB - Clifford product in the Clifford algebra Cl(B1)

### Calling Sequence:

cmulB(p1,p2,B1);

### Parameters:

p1, p2 - expressions of the type 'cliscalar' or 'clipolynom'

B1 - matrix of a bilinear form B1

### Description:

- Procedure 'cmulB' gives the Clifford product in the Clifford algebra Cl(B1) where B1 an arbitrary bilinear form.
- It differs from [cmul](#) in that a user may enter any form B1 as the third argument and the product of p1 and p2 will be computed in Cl(B1) regardless of the current value of B. Recall that B, when defined, is a global variable storing a bilinear form B used as a default to compute the Clifford product in Cl(B). Thus, 'cmul' by default computes the Clifford product with the form B while 'cmulB' computes the same product with the user-supplied form B1.
- The bilinear form B1 is totally arbitrary. It may be symbolic, undefined, symmetric, diagonal, with or without an antisymmetric part, numeric. When used as the third argument, it does not overwrite B.
- No infix form for 'cmulB' is available since it is intended to be used with the third argument. If the third argument is not entered, an error message is displayed.
- Calls to 'cmulB' with different values of B1 are made when computing products of tensors with [GTP:-gradedprod](#) or [GTP:-gprod](#) represented as polynomials of type [GTP:-`type/gradedpolynom`](#), [GTP:-`type/gradedmonom`](#), or [GTP:-`type/tensorprod`](#). In that case, B1 is expected to be diagonal. See examples below.

### Examples:

```
[ > restart:with(Clifford):with(GTP):
[ > B:=linalg[diag](1,1):B1:=linalg[diag](1,-1):B2:=linalg[matrix](2,2,[1,-1,1,-1]):
[ > p1:=e1-2*e2+e1we2:p2:=-e1we2+3*e2:
[ > cmul(p1,p2);
Cliplus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type
?cliprod for help.
                                -e2 + e1 - 5 Id + 3 e1we2
[ > cmulB(p1,p2); #testing an error message
Error, (in GTP:-cmulB) GTP:-cmulB uses a 3rd argument, B1 (of type matrix), which is missing
[ > cmulB(p1,p2,B); #product of p1 and p2 in Cl(B)
                                -e2 + e1 - 5 Id + 3 e1we2
[ > cmulB(p1,p2,B1); #product of p1 and p2 in Cl(B1)
                                -e2 - e1 + 5 Id + 3 e1we2
[ > cmulB(p1,p2,B2); #product of p1 and p2 in Cl(B2)
                                4 e2 - 2 e1 + e1we2 + 3 Id
[ The above products can also be accomplished using procedure Clifford:-cmul with optional index. For example,
[ > cmul[B](p1,p2); #product of p1 and p2 in Cl(B)
                                e1 - e2 + 3 e1we2 - 5 Id
[ > cmul[B1](p1,p2); #product of p1 and p2 in Cl(B1)
                                -e1 - e2 + 3 e1we2 + 5 Id
[ > cmul[B2](p1,p2); #product of p1 and p2 in Cl(B2)
                                -2 e1 + 4 e2 + e1we2 + 3 Id
[ > B3:=matrix(2,2,[]): #matrix B3 may be symbolic
[ > cmulB(p1,p2,B3);
(B31,2 + B32,2) e1 - (B31,1 - 2 B32,1 + 3 B31,2) e2 + (3 - B32,1 + B31,2) e1we2
+ (B32,2 B31,1 - B32,1 B31,2 + 3 B31,2 - 6 B32,2) Id
```

Here is an example where 'cmulB' will be called by [GTP:-gradedprod](#), [GTP:-gprod](#) when three different arguments B, B1, and B2

are used. Remember that in that case all three forms B, B1, and B2 must be diagonal.

```
> gprod(3*e1 &t e2 &t e2, 2*e1 &t e2 &t e2,B,B1,B1);  
-6((Id &t Id) &t Id)  
> gradedprod((2/3)*(e1 &t e2)-(5/a)*(e2we1 &t e2we1),3*(e1 &t e2)+Pi*(e1we2 &t  
e2we1),B1,B);
```

$$-2 (Id \&t Id) + \frac{2}{3} \pi (e2 \&t e1) + \frac{15 (e2 \&t e1)}{a} - \frac{5 \pi (Id \&t Id)}{a}$$

```
> gcollect(%);
```

$$-\frac{(2a + 5\pi)(Id \&t Id)}{a} + \frac{1}{3}(2\pi a + 45)(e2 \&t e1)$$

```
> evalm(B); #B is not overwritten
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
>
```

```
>
```

**See Also:** [Clifford:-cmul](#), [GTP:-`type/gradedmonom`](#), [GTP:-gbasis](#), [GTP:-`type/gradedodd`](#), [GTP:-grade](#), [GTP:-`&t`](#), [Clifford:-`type/tensorprod`](#), [GTP:-gradedprod](#), [GTP:-gprod](#), [GTP:-`type/gradedeven`](#)

Last revised: November 5, 2002/RA

**Function:** GTP:-gbasis - define a standard basis in the graded tensor product of n Clifford algebras Cl(B1), Cl(B2),..., Cl(Bn)

### Calling Sequence:

```
gbasis(L1,L2,...,Ln);  
gbasis(L$N);
```

### Parameters:

L1, L2,...,Ln - lists of basis elements in Cl(B1), Cl(B2),..., Cl(Bn)  
L - one list of basis elements in Cl(B) replicated n times in the call gbasis(L\$N)  
n - rank of the graded tensor product of Cl(B) &t Cl(B) &t ... &t Cl(B), n>=2

### Description:

- Procedure 'gbasis' writes a basis for a graded tensor product [GTP:-`&t`](#) of n Clifford algebras Cl(B1), Cl(B2),..., Cl(Bn) where each Clifford algebra Cl(Bi) is over a real vector space Vi endowed with a bilinear form Bi.
- The input may consist of a sequence of n lists L1, L2,...,Ln where each list Li, 1<=i<=n, contains a canonical basis for Cl(Bi) obtained with a help of [Clifford:-cbasis](#). Note that n must be at least 2.
- If all lists are the same, then a shorter input is possible of the form 'L\$N' where L is a basis for Cl(B) and n is the rank of the desired graded tensor product.
- It returns a list of elements of type [GTP:-`type/tensorprod`](#).
- For more information, see also [makealiases](#).

### Examples:

```
> restart:with(Clipford):with(GTP):eval(makealiases(4)):  
> L:=cbasis(2);L1:=cbasis(2);L2:=cbasis(3);  
                                     L := [Id, e1, e2, e12]  
                                     L1 := [Id, e1, e2, e12]  
                                     L2 := [Id, e1, e2, e3, e12, e13, e23, e123]  
> gbasis(L$2);nops(%);  
Cliplus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type  
?cliprod for help.  
[Id &t Id, Id &t e1, Id &t e2, Id &t e12, e1 &t Id, e1 &t e1, e1 &t e2, e1 &t e12, e2 &t Id, e2 &t e1, e2 &t e2, e2 &t e12,  
 e12 &t Id, e12 &t e1, e12 &t e2, e12 &t e12]  
                                     16  
> gbasis(L1,L2);nops(%);  
[Id &t Id, Id &t e1, Id &t e2, Id &t e3, Id &t e12, Id &t e13, Id &t e23, Id &t e123, e1 &t Id, e1 &t e1, e1 &t e2, e1 &t e3,  
 e1 &t e12, e1 &t e13, e1 &t e23, e1 &t e123, e2 &t Id, e2 &t e1, e2 &t e2, e2 &t e3, e2 &t e12, e2 &t e13, e2 &t e23,  
 e2 &t e123, e12 &t Id, e12 &t e1, e12 &t e2, e12 &t e3, e12 &t e12, e12 &t e13, e12 &t e23, e12 &t e123]  
                                     32  
> gbasis(L); #testing an error message  
Error, (in GTP:-gbasis) at least two lists with elements of type 'clibasmon' or 'tensorprod' are needed  
as input  
> gbasis(L$3);nops(%);  
[(Id &t Id) &t Id, (Id &t Id) &t e1, (Id &t Id) &t e2, (Id &t Id) &t e12, (Id &t e1) &t Id, (Id &t e1) &t e1,  
 (Id &t e1) &t e2, (Id &t e1) &t e12, (Id &t e2) &t Id, (Id &t e2) &t e1, (Id &t e2) &t e2, (Id &t e2) &t e12,  
 (Id &t e12) &t Id, (Id &t e12) &t e1, (Id &t e12) &t e2, (Id &t e12) &t e12, (e1 &t Id) &t Id, (e1 &t Id) &t e1,  
 (e1 &t Id) &t e2, (e1 &t Id) &t e12, (e1 &t e1) &t Id, (e1 &t e1) &t e1, (e1 &t e1) &t e2, (e1 &t e1) &t e12,  
 (e1 &t e2) &t Id, (e1 &t e2) &t e1, (e1 &t e2) &t e2, (e1 &t e2) &t e12, (e1 &t e12) &t Id, (e1 &t e12) &t e1,  
 (e1 &t e12) &t e2, (e1 &t e12) &t e12, (e2 &t Id) &t Id, (e2 &t Id) &t e1, (e2 &t Id) &t e2, (e2 &t Id) &t e12,  
 (e2 &t e1) &t Id, (e2 &t e1) &t e1, (e2 &t e1) &t e2, (e2 &t e1) &t e12, (e2 &t e2) &t Id, (e2 &t e2) &t e1,  
 (e2 &t e2) &t e2, (e2 &t e2) &t e12, (e2 &t e12) &t Id, (e2 &t e12) &t e1, (e2 &t e12) &t e2, (e2 &t e12) &t e12,
```

$(e_{12} \wedge Id) \wedge Id, (e_{12} \wedge Id) \wedge e_1, (e_{12} \wedge Id) \wedge e_2, (e_{12} \wedge Id) \wedge e_{12}, (e_{12} \wedge e_1) \wedge Id, (e_{12} \wedge e_1) \wedge e_1,$   
 $(e_{12} \wedge e_1) \wedge e_2, (e_{12} \wedge e_1) \wedge e_{12}, (e_{12} \wedge e_2) \wedge Id, (e_{12} \wedge e_2) \wedge e_1, (e_{12} \wedge e_2) \wedge e_2, (e_{12} \wedge e_2) \wedge e_{12},$   
 $(e_{12} \wedge e_{12}) \wedge Id, (e_{12} \wedge e_{12}) \wedge e_1, (e_{12} \wedge e_{12}) \wedge e_2, (e_{12} \wedge e_{12}) \wedge e_{12}]$

64

[ >

[ >

**See Also:** [Clifford:-cbasis](#), [GTP:-`type/gradedmonom`](#), [GTP:-gbasis](#), [GTP:-`&t`](#), [Clifford:-`type/tensorprod`](#)

Last revised: November 5, 2002/RA

**- Function:** GTP:-gcollect - collect graded tensors with respect to the basis elements

**Calling Sequence:**

```
gcollect(p);  
gcollect(p,s);
```

**Parameters:**

p - polynomial of type [GTP:-`type/gradedpolynom`](#)  
s - (optional) string which could be one of [simplify](#), [normal](#), etc.,

**- Description:**

- Procedure 'gcollect' collects terms in a graded polynomial with respect to its basis elements of [Clifford:-`type/tensorprod`](#).
- When the optional string is used, appropriate operation is performed on the coefficients. See [collect](#) for more help.
- By default, coefficients are factored.

**- Examples:**

```
[ > restart:with(Clifford):with(GTP):  
[ > p:=(a/b-a^2/b^2)*(e1 &t e2) + 2*(&t(e1,e2));gcollect(p);gcollect(p,simplify);  
  
p :=  $\left(\frac{a}{b} - \frac{a^2}{b^2}\right)(e1 \&t e2) + 2(e1 \&t e2)$   
       $-\frac{(a+b)(a-2b)(e1 \&t e2)}{b^2}$   
       $-\frac{(-ab+a^2-2b^2)(e1 \&t e2)}{b^2}$   
[ > p:=-2*a^3*&t(Id,Id,Id)+32*a*b^2*&t(Id,Id,Id);  
      p :=  $-2a^3((Id \&t Id) \&t Id) + 32ab^2((Id \&t Id) \&t Id)$   
[ > gcollect(p);gcollect(p,normal);gcollect(p,simplify);  
       $-2a(a-4b)(a+4b)((Id \&t Id) \&t Id)$   
       $(-2a^3+32ab^2)((Id \&t Id) \&t Id)$   
       $(-2a^3+32ab^2)((Id \&t Id) \&t Id)$   
[ >  
[ >
```

**- See Also:** [Clifford:-`type/tensorprod`](#), [GTP:-`type/gradedmonom`](#), [GTP:-`type/gradedpolynom`](#), [GTP:-`&t`](#)

**Function:** `GTP:-gprod` - compute a graded product of elements of the type 'tensorprod' or 'gradedmonom'

### Calling Sequence:

```
gprod(p1,p2);  
gprod(p1,p2,B1,B2,...,Br);
```

### Parameters:

`p1, p2` - graded monomials of type [GTP:-`type/gradedmonom`](#) or [Clifford:-`type/tensorprod`](#) of rank `r`  
`B1,B2,...,Br` - (optional) sequence of `r` diagonal forms `Bi`,  $1 \leq i \leq r$ , where `r` is the tensor rank of `p1` and `p2`

### Description:

- Procedure 'gprod' computes a graded product of two monomials in the graded tensor product  $Cl(B1) \&t Cl(B2) \&t \dots \&t Cl(Br)$  of `r` Clifford algebras  $Cl(B1), Cl(B2), \dots, Cl(Br)$ .
- When the optional sequence is used, Clifford products are computed component-wise in  $Cl(B1), Cl(B2), \dots, Cl(Br)$  with a help of the procedure [GTP:-cmulB](#). However, the  $\mathbb{Z}_2$ -gradation is taken into consideration in order to assure that elements of the type `e1` &t 1 and 1 &t `e1` belonging to  $Cl(B1) \&t Cl(B2)$  anticommute.
- Note that the product of two homogeneous elements `a` &t `b` and `c` &t `d` of  $Cl(B1) \&t Cl(B2)$  is computed as follows:

$$\text{gprod}(a \&t b, c \&t d) = (-1)^{(\text{grade}(b) * \text{grade}(c))} * (\text{cmulB}(a,c,B1) \&t \text{cmulB}(b,d,B2))$$

where `a` and `c` are homogeneous elements that belong to  $Cl(B1)$ , `b` and `d` are homogeneous elements that belong to  $Cl(B2)$ , and the grades of `b` and `c` modulo 2 may be computed with [GTP:-grade](#).

- Since `b` and `c` elements are homogeneous elements, they are either even or odd. If `b` is even, then  $\text{grade}(b) = 0 \pmod{2}$ , otherwise  $\text{grade}(b) = 1 \pmod{2}$ .
- The graded multiplication defined above on homogeneous tensors of rank 2 may be extended to homogeneous tensors of higher ranks and then to non-homogeneous tensors by means of linearity. In that latter case, use procedure [GTP:-gradedprod](#).
- When the optional sequence is not used, the default bilinear form `B` is applied. Thus, in this case, the `r` products will be computed in `r` different copies of  $Cl(B)$ .
- The ranks of `p1` and `p2` must be the same. They can be found with [GTP:-`type/tensorrank`](#).

### Examples:

```
[ > restart:with(Clifford):with(GTP):eval(makealiases(5)):_prolevel:=true:  
[ Example 1:  
[ > B:=linalg[diag](1,-1,-1,-1):B1:=linalg[diag](-1,1,-1,1):B2:=linalg[diag](1,1,-1,1):  
[ > type(e1 &t e2,tensorprod),  
[ type(2*(e1 &t e3),gradedmonom),  
[ type(2*&t(e1,e2,e3),gradedmonom);  
Cliplus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type  
?cliprod for help.  
  
true, true, true  
[ > gprod(1 &t e1,e2 &t 1);gprod(e2 &t 1,1 &t e1);  
-(e2 &t e1)  
e2 &t e1  
[ > gprod(Id &t e1 &t e2,e1 &t e2 &t Id);tensorrank(%)  
-((e1 &t e12) &t e2)  
3  
[ > gprod(e1 &t e2, e3 &t e1,B1,B2);  
e13 &t e12  
[ > gprod(Id &t e2 &t e2,e1 &t Id &t e1);tensorrank(%)  
-((e1 &t e2) &t e12)  
3  
[ > gprod(3*(Id &t e2),2*(e1 &t Id));tensorrank(%)
```

$$-6(e_1 \otimes e_2)$$

2

```
> gprod(&t(1,e1,e2),&t(1,e1,e2));
```

$$(Id \otimes Id) \otimes Id$$

**Example 2:** Consider a graded tensor product of three copies of the Clifford algebra  $Cl(B)$  where  $B$  is a quadratic form  $(1,-1,-1,-1)$ . We will show that the above definition of the product assures non-commutativity of the basis elements.

```
> B:=linalg[diag](1,-1,-1,-1):
```

```
> for i from 1 to 4 do E[i]:=&t(e| |i,1,1) od; #basis 1-vectors in one copy of Cl(B)
```

$$E_1 := (e_1 \otimes 1) \otimes 1$$

$$E_2 := (e_2 \otimes 1) \otimes 1$$

$$E_3 := (e_3 \otimes 1) \otimes 1$$

$$E_4 := (e_4 \otimes 1) \otimes 1$$

```
> for i from 1 to 4 do F[i]:=&t(1,e| |i,1) od; #basis 1-vectors in another copy of Cl(B)
```

$$F_1 := (1 \otimes e_1) \otimes 1$$

$$F_2 := (1 \otimes e_2) \otimes 1$$

$$F_3 := (1 \otimes e_3) \otimes 1$$

$$F_4 := (1 \otimes e_4) \otimes 1$$

```
> for i from 1 to 4 do G[i]:=&t(1,1,e| |i) od; #basis 1-vectors in another copy of Cl(B)
```

$$G_1 := (1 \otimes 1) \otimes e_1$$

$$G_2 := (1 \otimes 1) \otimes e_2$$

$$G_3 := (1 \otimes 1) \otimes e_3$$

$$G_4 := (1 \otimes 1) \otimes e_4$$

We will combine now all basis rank 1-tensors into a list A. These rank 1-tensors generate the graded tensor product of rank three of the three copies of  $Cl(B)$ . This new algebra is  $\mathbb{Z}_2$ -isomorphic with  $Cl(3B)$  where  $3B$  is a quadratic form of signature  $(3,9)$ .

Below we explicitly show that the generators anticommute.

```
> printlevel:=2:A:=[seq(E[i],i=1..4),seq(F[i],i=1..4),seq(G[i],i=1..4)];
```

```
A := [(e1 &t 1) &t 1, (e2 &t 1) &t 1, (e3 &t 1) &t 1, (e4 &t 1) &t 1, (1 &t e1) &t 1, (1 &t e2) &t 1, (1 &t e3) &t 1,
(1 &t e4) &t 1, (1 &t 1) &t e1, (1 &t 1) &t e2, (1 &t 1) &t e3, (1 &t 1) &t e4]
```

```
> for i from 1 to nops(A) do
```

```
  for j from i to nops(A) do
```

```
    [i,j],[A[i],A[j]],(1/2)*(gprod(A[i],A[j])+gprod(A[j],A[i]))
```

```
  od od;
```

$$[1, 1], [(e_1 \otimes 1) \otimes 1, (e_1 \otimes 1) \otimes 1], (Id \otimes Id) \otimes Id$$

$$[1, 2], [(e_1 \otimes 1) \otimes 1, (e_2 \otimes 1) \otimes 1], 0$$

$$[1, 3], [(e_1 \otimes 1) \otimes 1, (e_3 \otimes 1) \otimes 1], 0$$

$$[1, 4], [(e_1 \otimes 1) \otimes 1, (e_4 \otimes 1) \otimes 1], 0$$

$$[1, 5], [(e_1 \otimes 1) \otimes 1, (1 \otimes e_1) \otimes 1], 0$$

$$[1, 6], [(e_1 \otimes 1) \otimes 1, (1 \otimes e_2) \otimes 1], 0$$

$$[1, 7], [(e_1 \otimes 1) \otimes 1, (1 \otimes e_3) \otimes 1], 0$$

$$[1, 8], [(e_1 \otimes 1) \otimes 1, (1 \otimes e_4) \otimes 1], 0$$

$$[1, 9], [(e_1 \otimes 1) \otimes 1, (1 \otimes 1) \otimes e_1], 0$$

$$[1, 10], [(e_1 \otimes 1) \otimes 1, (1 \otimes 1) \otimes e_2], 0$$

$$[1, 11], [(e_1 \otimes 1) \otimes 1, (1 \otimes 1) \otimes e_3], 0$$

$$[1, 12], [(e_1 \otimes 1) \otimes 1, (1 \otimes 1) \otimes e_4], 0$$

$$[2, 2], [(e_2 \otimes 1) \otimes 1, (e_2 \otimes 1) \otimes 1], -(Id \otimes Id) \otimes Id$$

$$[2, 3], [(e_2 \otimes 1) \otimes 1, (e_3 \otimes 1) \otimes 1], 0$$

$$[2, 4], [(e_2 \otimes 1) \otimes 1, (e_4 \otimes 1) \otimes 1], 0$$

[2, 5], [(e2 & t 1) & t 1, (1 & t e1) & t 1], 0  
 [2, 6], [(e2 & t 1) & t 1, (1 & t e2) & t 1], 0  
 [2, 7], [(e2 & t 1) & t 1, (1 & t e3) & t 1], 0  
 [2, 8], [(e2 & t 1) & t 1, (1 & t e4) & t 1], 0  
 [2, 9], [(e2 & t 1) & t 1, (1 & t 1) & t e1], 0  
 [2, 10], [(e2 & t 1) & t 1, (1 & t 1) & t e2], 0  
 [2, 11], [(e2 & t 1) & t 1, (1 & t 1) & t e3], 0  
 [2, 12], [(e2 & t 1) & t 1, (1 & t 1) & t e4], 0  
 [3, 3], [(e3 & t 1) & t 1, (e3 & t 1) & t 1], -((Id & t Id) & t Id)  
 [3, 4], [(e3 & t 1) & t 1, (e4 & t 1) & t 1], 0  
 [3, 5], [(e3 & t 1) & t 1, (1 & t e1) & t 1], 0  
 [3, 6], [(e3 & t 1) & t 1, (1 & t e2) & t 1], 0  
 [3, 7], [(e3 & t 1) & t 1, (1 & t e3) & t 1], 0  
 [3, 8], [(e3 & t 1) & t 1, (1 & t e4) & t 1], 0  
 [3, 9], [(e3 & t 1) & t 1, (1 & t 1) & t e1], 0  
 [3, 10], [(e3 & t 1) & t 1, (1 & t 1) & t e2], 0  
 [3, 11], [(e3 & t 1) & t 1, (1 & t 1) & t e3], 0  
 [3, 12], [(e3 & t 1) & t 1, (1 & t 1) & t e4], 0  
 [4, 4], [(e4 & t 1) & t 1, (e4 & t 1) & t 1], -((Id & t Id) & t Id)  
 [4, 5], [(e4 & t 1) & t 1, (1 & t e1) & t 1], 0  
 [4, 6], [(e4 & t 1) & t 1, (1 & t e2) & t 1], 0  
 [4, 7], [(e4 & t 1) & t 1, (1 & t e3) & t 1], 0  
 [4, 8], [(e4 & t 1) & t 1, (1 & t e4) & t 1], 0  
 [4, 9], [(e4 & t 1) & t 1, (1 & t 1) & t e1], 0  
 [4, 10], [(e4 & t 1) & t 1, (1 & t 1) & t e2], 0  
 [4, 11], [(e4 & t 1) & t 1, (1 & t 1) & t e3], 0  
 [4, 12], [(e4 & t 1) & t 1, (1 & t 1) & t e4], 0  
 [5, 5], [(1 & t e1) & t 1, (1 & t e1) & t 1], (Id & t Id) & t Id  
 [5, 6], [(1 & t e1) & t 1, (1 & t e2) & t 1], 0  
 [5, 7], [(1 & t e1) & t 1, (1 & t e3) & t 1], 0  
 [5, 8], [(1 & t e1) & t 1, (1 & t e4) & t 1], 0  
 [5, 9], [(1 & t e1) & t 1, (1 & t 1) & t e1], 0  
 [5, 10], [(1 & t e1) & t 1, (1 & t 1) & t e2], 0  
 [5, 11], [(1 & t e1) & t 1, (1 & t 1) & t e3], 0  
 [5, 12], [(1 & t e1) & t 1, (1 & t 1) & t e4], 0  
 [6, 6], [(1 & t e2) & t 1, (1 & t e2) & t 1], -((Id & t Id) & t Id)  
 [6, 7], [(1 & t e2) & t 1, (1 & t e3) & t 1], 0  
 [6, 8], [(1 & t e2) & t 1, (1 & t e4) & t 1], 0  
 [6, 9], [(1 & t e2) & t 1, (1 & t 1) & t e1], 0  
 [6, 10], [(1 & t e2) & t 1, (1 & t 1) & t e2], 0  
 [6, 11], [(1 & t e2) & t 1, (1 & t 1) & t e3], 0  
 [6, 12], [(1 & t e2) & t 1, (1 & t 1) & t e4], 0  
 [7, 7], [(1 & t e3) & t 1, (1 & t e3) & t 1], -((Id & t Id) & t Id)  
 [7, 8], [(1 & t e3) & t 1, (1 & t e4) & t 1], 0  
 [7, 9], [(1 & t e3) & t 1, (1 & t 1) & t e1], 0  
 [7, 10], [(1 & t e3) & t 1, (1 & t 1) & t e2], 0  
 [7, 11], [(1 & t e3) & t 1, (1 & t 1) & t e3], 0

```

[7, 12], [(1 & e3) & 1, (1 & 1) & e4], 0
[8, 8], [(1 & e4) & 1, (1 & e4) & 1], -((Id & Id) & Id)
[8, 9], [(1 & e4) & 1, (1 & 1) & e1], 0
[8, 10], [(1 & e4) & 1, (1 & 1) & e2], 0
[8, 11], [(1 & e4) & 1, (1 & 1) & e3], 0
[8, 12], [(1 & e4) & 1, (1 & 1) & e4], 0
[9, 9], [(1 & 1) & e1, (1 & 1) & e1], (Id & Id) & Id
[9, 10], [(1 & 1) & e1, (1 & 1) & e2], 0
[9, 11], [(1 & 1) & e1, (1 & 1) & e3], 0
[9, 12], [(1 & 1) & e1, (1 & 1) & e4], 0
[10, 10], [(1 & 1) & e2, (1 & 1) & e2], -((Id & Id) & Id)
[10, 11], [(1 & 1) & e2, (1 & 1) & e3], 0
[10, 12], [(1 & 1) & e2, (1 & 1) & e4], 0
[11, 11], [(1 & 1) & e3, (1 & 1) & e3], -((Id & Id) & Id)
[11, 12], [(1 & 1) & e3, (1 & 1) & e4], 0
[12, 12], [(1 & 1) & e4, (1 & 1) & e4], -((Id & Id) & Id)

```

Thus, we have

$$(1/2) * (\text{gprod}(A[i], A[j]) + \text{gprod}(A[j], A[i])) = 0, 1 \leq i, j \leq 12, i < j$$

and

$$\text{gprod}(A[i], A[i]) = ((Id \& Id) \& Id) \text{ or } \text{gprod}(A[i], A[i]) = -((Id \& Id) \& Id), i = 1..12.$$

```

> for i from 1 to nops(A) do i, A[i], gprod(A[i], A[i]) od;
1, (e1 & 1) & 1, (Id & Id) & Id
2, (e2 & 1) & 1, -((Id & Id) & Id)
3, (e3 & 1) & 1, -((Id & Id) & Id)
4, (e4 & 1) & 1, -((Id & Id) & Id)
5, (1 & e1) & 1, (Id & Id) & Id
6, (1 & e2) & 1, -((Id & Id) & Id)
7, (1 & e3) & 1, -((Id & Id) & Id)
8, (1 & e4) & 1, -((Id & Id) & Id)
9, (1 & 1) & e1, (Id & Id) & Id
10, (1 & 1) & e2, -((Id & Id) & Id)
11, (1 & 1) & e3, -((Id & Id) & Id)
12, (1 & 1) & e4, -((Id & Id) & Id)

```

We have explicitly shown that the generators  $A[1]$ ,  $A[5]$  and  $A[9]$  have squares equal to +1 while the remaining generators have squares equal to -1.

```

> map(grade, A); #all elements of A are of grade 1
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
> map(tensrank, A); #all elements of A are tensors of rank 3
[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
>
>

```

**See Also:** [GTP:-`type/gradedmonom`](#), [GTP:-`gbasis`](#), [GTP:-`type/gradedodd`](#), [GTP:-`grade`](#), [GTP:-`&t`](#), [Clifford:-`type/tensorprod`](#), [GTP:-`gradedprod`](#), [GTP:-`type/gradedeven`](#)



**Function:** `GTP:-grade` - find the grade of any element of type 'cliscalar', 'climon', 'tensorprod', 'gradedmonom'

**Calling Sequence:**

`grade(p);`

**Parameters:**

p - element of one of these types: [Clifford:-`type/cliscalar`](#), [Clifford:-`type/climon`](#), [Clifford:-`type/tensorprod`](#), [GTP:-`type/gradedmonom`](#).

**Description:**

- Procedure 'grade' returns 0 if p is even and 1 if p is odd.
- When applied to homogeneous elements of the type [GTP:-`type/tensorprod`](#) or [GTP:-`type/gradedmonom`](#), the procedure adds the grades of individual entries. For example, the grade of `e1 &t e1` is 0 since `e1` and `e2` are odd elements, the grade of `e1we2 &t e1 &t e2` is 0, while the grade of `e1 &t Id` is 1.
- Check also [GTP:-`type/gradedodd`](#) and [GTP:-`type/gradedeven`](#).

**Examples:**

```
[ > restart:with(Clifford):with(GTP):
[ > grade(e1 &t e1),grade(e1we2 &t e1 &t e2),grade(e1 &t Id);
Clipus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type
?cliprod for help.
                                0, 0, 1
[ > grade(2*e1we2);
                                0
[ > grade(1);
                                0
[ > grade(e1we2we3);
                                1
[ > grade(e1 &t e1we2 &t e3);
                                0
[ > grade(&t(e1,e2,e3,e4));
                                0
[ >
[ >
```

**See Also:** [GTP:-`type/gradedmonom`](#), [GTP:-gbasis](#), [GTP:-`type/gradedodd`](#), [GTP:-grade](#), [GTP:-`&t`](#), [Clifford:-`type/tensorprod`](#), [GTP:-gradedprod](#), [GTP:-`type/tensorrank`](#), [GTP:-`type/gradedeven`](#)

Last revised: November 5, 2002/RA

**Function:** `GTP:-`type/gradedeven`` - define type 'gradedeven'

### Calling Sequence:

`type(p,gradedeven);`

### Parameters:

p - element of one of these types: [GTP:-`type/gradedpolynom`](#), [GTP:-`type/gradedmonom`](#), [Clifford:-`type/tensorprod`](#)

### Description:

- Polynomial elements in the graded tensor product  $Cl(B_1) \&t Cl(B_2) \&t \dots \&t Cl(B_r)$  of  $r$  Clifford algebras  $Cl(B_i)$ , where  $B_i$  are quadratic forms,  $1 \leq i \leq r$ , whose all monomial terms are of grade 0 are by definition of type 'gradedeven'. Otherwise, if at least one monomial term is of grade 1, the entire polynomial is of type 'gradedodd'.
- Polynomials of type 'gradedeven' form a subalgebra in the graded tensor algebra  $Cl(B_1) \&t Cl(B_2) \&t \dots \&t Cl(B_r)$  which is isomorphic with the even subalgebra of a Clifford algebra  $Cl(B)$  where  $B$  is an orthogonal sum of the quadratic forms  $B_1, B_2, \dots, B_r$ .
- See also [GTP:-`type/gradedodd`](#) and [GTP:-grade](#).

### Examples:

```
[ > restart:with(Clifford):with(GTP):eval(makealiases(5)):_prolevel:=true:
[ Example 1: Some type checking.
[ > p:=e1 &t e1 + 2* (e1we2 &t e2);
  Cliplus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type
  ?cliprod for help.
                                     p := (e1 &t e1) + 2 (e12 &t e2)
[ > type(p,gradedeven);
                                     false
[ > B:=linalg[diag](1,1,1):
  p1:=2*&t(e1we2,e2we3)-Pi*&t(e1,e2);p2:=2*(e1we2 &t (Id-e1we3+e2we3));
                                     p1 := 2 (e12 &t e23) - pi (e1 &t e2)
                                     p2 := 2 (e12 &t Id) - 2 (e12 &t e13) + 2 (e12 &t e23)
[ > type(p1,gradedeven),type(p2,gradedeven);
                                     true, true
[ > type(gradedprod(p1,p2),gradedeven);
                                     true
[ Example 2: Even basis monomials form a basis for an even subalgebra of  $Cl(B_1) \&t Cl(B_2) \&t \dots \&t Cl(B_r)$ , while the odd
[ basis monomials give an even basis monomial when multiplied out.
[ > dim:=2:K:=cbasis(2):L:=gbasis(K$2);
  L := [Id &t Id, Id &t e1, Id &t e2, Id &t e12, e1 &t Id, e1 &t e1, e1 &t e2, e1 &t e12, e2 &t Id, e2 &t e1, e2 &t e2, e2 &t e12,
  e12 &t Id, e12 &t e1, e12 &t e2, e12 &t e12]
[ > evenmonomials:=select(type,L,gradedeven);
  evenmonomials := [Id &t Id, Id &t e12, e1 &t e1, e1 &t e2, e2 &t e1, e2 &t e2, e12 &t Id, e12 &t e12]
[ > oddmonomials:=select(type,L,gradedodd);
  oddmonomials := [Id &t e1, Id &t e2, e1 &t Id, e1 &t e12, e2 &t Id, e2 &t e12, e12 &t e1, e12 &t e2]
[ We now compute all possible products of even and odd monomials, and store them in Se and So respectively. Then we check that
[ the elements of Se and So are plus or minus the elements from 'evenmonomials'. In Seo we will store all products of even and odd
[ elements. These elements should be either plus or minus the elements from 'oddmonomials':
[ > Se:={}:So:={}:Seo:={}:
  for i from 1 to nops(evenmonomials) do
  for j from 1 to nops(evenmonomials) do
    Se:=Se union {gprod(evenmonomials[i],evenmonomials[j])}:
    So:=So union {gprod(oddmonomials[i],oddmonomials[j])}:
    Seo:=Seo union {gprod(oddmonomials[i],evenmonomials[j])}:
    Seo:=Seo union {gprod(evenmonomials[i],oddmonomials[j])}:
[ ]
```

```

[ end do end do:
[ > Se,nops(Se);
[ {e1 &t e2, e12 &t Id, -(e1 &t e1), e2 &t e2, Id &t Id, -(e2 &t e2), -(e12 &t Id), e1 &t e1, -(e2 &t e1), -(Id &t Id),
[   Id &t e12, e2 &t e1, -(e12 &t e12), -(Id &t e12), e12 &t e12, -(e1 &t e2)}, 16
[ > So,nops(So);
[ {e1 &t e2, e12 &t Id, -(e1 &t e1), e2 &t e2, Id &t Id, -(e2 &t e2), -(e12 &t Id), e1 &t e1, -(e2 &t e1), -(Id &t Id),
[   Id &t e12, e2 &t e1, -(e12 &t e12), -(Id &t e12), e12 &t e12, -(e1 &t e2)}, 16
[ > Seo,nops(Seo);
[ {-(e2 &t Id), -(Id &t e2), -(e2 &t e12), e12 &t e2, -(Id &t e1), -(e1 &t Id), Id &t e1, e1 &t Id, Id &t e2, e2 &t Id,
[   e1 &t e12, e12 &t e1, e2 &t e12, -(e1 &t e12), -(e12 &t e1), -(e12 &t e2)}, 16
[ Now we verify that elements in Se and So are of type 'gradedeven' while the elements in Seo are of type 'gradedodd':
[ > map(type,Se,gradedeven);
[
[                                     {true}
[ > map(type,So,gradedeven);
[
[                                     {true}
[ > map(type,Seo,gradedodd);
[
[                                     {true}
[ >
[ >

```

**See Also:** [GTP:-`type/gradedmonom`](#), [GTP:-gbasis](#), [GTP:-`type/gradedodd`](#), [GTP:-grade](#), [GTP:-`&t`](#), [Clifford:-`type/tensorprod`](#)

Last revised: November 5, 2002/RA

**- Function:** `GTP:-`type/gradedmonom`` - define a type 'gradedmonom'

**Calling Sequence:**

`type(p,gradedmonom);`

**Parameters:**

p - element of one of these types: ``*``, function, algebraic.

**- Description:**

- Monomial (homogeneous) elements in the graded tensor product  $Cl(B_1) \otimes Cl(B_2) \otimes \dots \otimes Cl(B_r)$  of r Clifford algebras  $Cl(B_i)$ , where  $B_i$  are quadratic forms,  $1 \leq i \leq r$ , are by definition of type 'gradedmonom'. Thus, they are either of type [Clifford:-`type/tensorprod`](#) or they are products of two elements, one of type [Clifford:-`type/tensorprod`](#) and one of type [Clifford:-`type/cliscalar`](#).
- See also [GTP:-`type/gradedpolynom`](#).

**- Examples:**

```
[ > restart:with(Clipford):with(GTP):
[ > type(e1 &t e1,gradedmonom),type(Pi*(e1we2 &t e1 &t e2),gradedmonom);
Cliplus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type
?cliprod for help.
true, true
[ > type(2*&t(e1,e2,e3),gradedmonom);
true
[ > type(2*&t(e1,e2,e3)+e2we3 &t e2we1,gradedmonom);
false
[ > type(2*&t(e1,e2,e3)+e2we3 &t e2we1,gradedpolynom);
true
[ > dim:=2:K:=cbasis(2):L:=gbasis(K$2);
L := [Id &t Id, Id &t e1, Id &t e2, Id &t e1we2, e1 &t Id, e1 &t e1, e1 &t e2, e1 &t e1we2, e2 &t Id, e2 &t e1, e2 &t e2,
e2 &t e1we2, e1we2 &t Id, e1we2 &t e1, e1we2 &t e2, e1we2 &t e1we2]
[ > map(type,L,gradedmonom);
[ true, true, true, true, true, true, true, true, true, true, true, true, true, true ]
[ >
[ >
```

**- See Also:** [GTP:-gbasis](#), [GTP:-`type/gradedodd`](#), [GTP:-grade](#), [GTP:-`&t`](#), [Clifford:-`type/tensorprod`](#), [GTP:-gradedprod](#), [GTP:-gprod](#), [GTP:-`type/gradedeven`](#)

**Function:** `GTP:-`type/gradedodd`` - define type 'gradedodd'

### Calling Sequence:

`type(p,gradedodd);`

### Parameters:

p - element of one of these types: [GTP:-`type/gradedpolynom`](#), [GTP:-`type/gradedmonom`](#), [Clifford:-`type/tensorprod`](#)

### Description:

- Polynomial elements in the graded tensor product  $Cl(B_1) \&t Cl(B_2) \&t \dots \&t Cl(B_r)$  of r Clifford algebras  $Cl(B_i)$ , where  $B_i$  are quadratic forms,  $1 \leq i \leq r$ , which contain all monomials of grade 1 are of type 'gradedodd'.
- Polynomials of type 'gradedodd' do not form a subalgebra in the graded tensor algebra  $Cl(B_1) \&t Cl(B_2) \&t \dots \&t Cl(B_r)$ . However, a product of two odd elements is even.
- See also [GTP:-`type/gradedeven`](#) and [GTP:-grade](#).

### Examples:

```
[ > restart:with(Clifford):with(GTP):
> p1:=e1 &t e1 &t e2+ 2* (e1 &t e2 &t e2);
Clipus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type
?cliprod for help.
                p1 := ((e1 &t e1) &t e2) + 2 ((e1 &t e2) &t e2)
> type(p1,gradedeven),type(p1,gradedodd);
                false, true
> p2:=2*&t(e1,e2,e3);type(p2,gradedodd);
                p2 := 2 ((e1 &t e2) &t e3)
                true
> B:=linalg[diag](1,1,1):type(gradedprod(p1,p2),gradedeven);
                true
> dim:=2:K:=cbasis(2):L:=gbasis(K$2);
L := [Id &t Id, Id &t e1, Id &t e2, Id &t e1we2, e1 &t Id, e1 &t e1, e1 &t e2, e1 &t e1we2, e2 &t Id, e2 &t e1, e2 &t e2,
      e2 &t e1we2, e1we2 &t Id, e1we2 &t e1, e1we2 &t e2, e1we2 &t e1we2]
> map(type,L,gradedodd);
                [false, true, true, false, true, false, false, true, true, false, false, true, false, true, true, false]
> map(type,L,gradedeven);
                [true, false, false, true, false, true, true, false, false, true, true, false, true, false, false, true]
>
```

**See Also:** [GTP:-`type/gradedmonom`](#), [GTP:-gbasis](#), [GTP:-grade](#), [GTP:-`&t`](#), [Clifford:-`type/tensorprod`](#), [GTP:-gradedprod](#), [GTP:-gprod](#), [GTP:-`type/gradedeven`](#)

Last revised: November 5, 2002/RA

**Function:** `GTP:-`type/gradedpolynom`` - define a type 'gradedpolynom'

**Calling Sequence:**

`type(p,gradedpolynom);`

**Parameters:**

p - element of one of these types: `\*`, `+`, 'function', or 'algebraic'.

**Description:**

- Polynomial elements in the graded tensor product  $Cl(B_1) \otimes Cl(B_2) \otimes \dots \otimes Cl(B_r)$  of r Clifford algebras  $Cl(B_i)$ , where  $B_i$  are quadratic forms,  $1 \leq i \leq r$ , are by definition of type 'gradedpolynom'. Thus, they are linear combinations of the basis elements of the type [Clifford:-`type/tensorprod`](#) while their coefficients are of the type [Clifford:-`type/cliscalar`](#).
- See also [GTP:-`type/gradedmonom`](#). Elements of type 'gradedmonom' are also of type 'gradedpolynom'.
- Elements of the type 'gradedpolynom' are multivariate polynomials used by the procedures [GTP:-gprod](#) and [GTP:-gradedprod](#).

**Examples:**

```
[ > restart:with(Clifford):with(GTP):
[ > type(e1 &t e1 + 2*(e1we2 &t e2),gradedpolynom),
  type(Pi*(e1we2 &t e1 &t e2),gradedpolynom);
Clipus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type
?cliprod for help.
                                     true, true
[ > type(2*&t(e1,e2,e3),gradedpolynom);
                                     true
[ > dim:=2:K:=cbasis(2):L:=gbasis(K$2);
L := [Id &t Id, Id &t e1, Id &t e2, Id &t e1we2, e1 &t Id, e1 &t e1, e1 &t e2, e1 &t e1we2, e2 &t Id, e2 &t e1, e2 &t e2,
      e2 &t e1we2, e1we2 &t Id, e1we2 &t e1, e1we2 &t e2, e1we2 &t e1we2]
[ > map(type,L,gradedpolynom);
      [true, true, true, true, true, true, true, true, true, true, true, true, true, true]
[ >
[ >
```

**See Also:** [GTP:-`type/gradedmonom`](#), [GTP:-gbasis](#), [GTP:-`type/gradedodd`](#), [GTP:-grade](#), [GTP:-`&t`](#), [Clifford:-`type/tensorprod`](#), [GTP:-gradedprod](#), [GTP:-gprod](#), [GTP:-`type/gradedeven`](#)

Last revised: November 5, 2002/RA

**Function:** GTP:-gradedprod - compute a graded product of elements of the type 'tensorprod', 'gradedmonom', or 'gradedpolynom'

### Calling Sequence:

```
gradedprod(p1,p2);
gradedprod(p1,p2,B1,B2,...,Br);
```

### Parameters:

p1, p2 - graded polynomials of type [GTP:-type/gradedpolynom](#) of rank r  
 B1,B2,...,Br - (optional) sequence of r diagonal forms Bi,  $1 \leq i \leq r$ , where r is the tensor rank of p1 and p2

### Description:

- Procedure 'gradedprod' is an extension of [GTP:-prod](#) and computes a product of two graded polynomials in the graded tensor product  $Cl(B1) \otimes Cl(B2) \otimes \dots \otimes Cl(Br)$  of r Clifford algebras  $Cl(B1), Cl(B2), \dots, Cl(Br)$ . In particular, it can handle also monomials.
- When the optional sequence is used, Clifford products are computed component-wise on homogeneous elements in  $Cl(B1), Cl(B2), \dots, Cl(Br)$  with a help of the procedure [GTP:-cmulB](#). However, the Z2-gradation is taken into consideration in order to assure, for example, that elements of the type  $e1 \otimes 1$  and  $1 \otimes e1$  belonging to  $Cl(B1) \otimes Cl(B2)$  anticommute.
- For more information how multiplication is defined on homogeneous elements in the graded tensor product  $Cl(B1) \otimes Cl(B2)$  of two Clifford algebras  $Cl(B1)$  and  $Cl(B2)$  see [GTP:-gprod](#). This definition is extended to tensors of higher ranks and then to non-homogeneous tensors by linearity.
- When the optional sequence is not used, the default bilinear form B is applied. Thus, in this case, the r products will be computed in r different copies of  $Cl(B)$ .
- The ranks of p1 and p2 must be the same. They can be found with [GTP:-tensorrank](#).

### Examples:

```
[ > restart:with(Clifford):with(GTP):eval(makealiases(5)):_prolevel:=true:
[ Example 1:
[ > B:=linalg[diag](1,-1,-1,-1):
[ > type(e1 &t e2,tensorprod),
  type(2*(e1 &t e3),gradedmonom),
  type(2*&t(e1,e2,e3),gradedmonom);
  type((e1 &t e2) + b*(e1 &t e2) +e2 &t e3,gradedpolynom);
Cliplus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type
?cliprod for help.

                                true, true, true
                                true
[ > gradedprod(e1 &t e2 &t e2,e1 &t e2 &t e2);tensorrank(%);
                                -((Id &t Id) &t Id)
                                3
[ > gradedprod((e1 + e2) &t e2 &t Id,e1 &t e2 &t Id);tensorrank(%);
                                ((Id &t Id) &t Id) - ((e12 &t Id) &t Id)
                                3
[ > p:=a*(Id &t Id &t e2)-3*(e1we2 &t e2 &t e2we1 )+b*4*(Id &t e2 &t e1)-
  Id &t e1we2 &t e1we2;
                                p := a((Id &t Id) &t e2) - 3((e12 &t e2) &t e21) + 4 b((Id &t e2) &t e1) - ((Id &t e12) &t e12)
[ > type(p,gradedpolynom);tensorrank(p);
                                true
                                3
[ > r1:=gradedprod(-a*p,2*p);
r1 := 2 a^3 ((Id &t Id) &t Id) + 12 a^2 ((e12 &t e2) &t e1) - 16 a^2 b ((Id &t e2) &t e12) + 16 a ((Id &t Id) &t Id)
- 48 a b ((e12 &t Id) &t e2) - 32 a b^2 ((Id &t Id) &t Id) + 16 a b ((Id &t e1) &t e2)
```

```
> tensorrank(r1);
```

3

The results from 'gradedprod' may be collected using [GTP:-gcollect](#):

```
> gcollect(r1);
```

$$\begin{aligned} & -16 a^2 b ((Id \& t e2) \& t e12) + 12 a^2 ((e12 \& t e2) \& t e1) - 48 a b ((e12 \& t Id) \& t e2) + 16 a b ((Id \& t e1) \& t e2) \\ & + 2 a (a^2 - 16 b^2 + 8) ((Id \& t Id) \& t Id) \end{aligned}$$

**Example 2:** In Example 1, r1 is an element of  $C(B) \& t Cl(B) \& t Cl(B)$  which is  $\mathbb{Z}_2$ -isomorphic with  $Cl(3B)$  where 3B is a quadratic form of signature (3,9). Now use different quadratic forms in each component. For example, consider p defined above as an element of the algebra  $Cl(B) \& t Cl(B1) \& t Cl(B2)$  where B1 and B2 are defined below.

```
> B1:=linalg[diag](-1,1,-1,1):B2:=linalg[diag](1,1,-1,1):
```

Now, we compute the graded product of p with p. Notice, that the result r2 is different than r1:

```
> r2:=gradedprod(p,p,B,B1,B2);
```

$$\begin{aligned} r2 := & a^2 ((Id \& t Id) \& t Id) + 6 a ((e12 \& t e2) \& t e1) + 8 a b ((Id \& t e2) \& t e12) - 10 ((Id \& t Id) \& t Id) \\ & - 24 b ((e12 \& t Id) \& t e2) - 16 b^2 ((Id \& t Id) \& t Id) + 8 b ((Id \& t e1) \& t e2) \end{aligned}$$

```
> gcollect(r2);
```

$$\begin{aligned} & 8 a b ((Id \& t e2) \& t e12) + 6 a ((e12 \& t e2) \& t e1) - 24 b ((e12 \& t Id) \& t e2) + 8 b ((Id \& t e1) \& t e2) \\ & + (a^2 - 16 b^2 - 10) ((Id \& t Id) \& t Id) \end{aligned}$$

**Example 3:** Some more computations.

```
> p:=gradedprod(e1 & t e2,e2 & t e1we2-2*Pi*cos(alpha)*e1we2 & t Id -(a+b)/(a-b)*e2we1 & t e1);
```

$$p := -(e12 \& t e1) - 2 \pi \cos(\alpha) (e2 \& t e2) - \frac{a (e2 \& t e12)}{a-b} - \frac{b (e2 \& t e12)}{a-b}$$

```
> gcollect(p);
```

$$-\frac{(a+b) (e2 \& t e12)}{a-b} - (e12 \& t e1) - 2 \pi \cos(\alpha) (e2 \& t e2)$$

```
> p1:=&t(-Pi*cos(alpha)*e1,e2-e2we1,Id,2*e1);
```

```
p2:=&t(Id,e1-3*(a-b)/(a+c)*e1we2,e2we1,1);
```

$$p1 := -2 \pi \cos(\alpha) (((e1 \& t e2) \& t Id) \& t e1) + 2 \pi \cos(\alpha) (((e1 \& t e21) \& t Id) \& t e1)$$

$$p2 := (((Id \& t e1) \& t e21) \& t 1) - \frac{3 (a-b) (((Id \& t e12) \& t e21) \& t 1)}{a+c}$$

```
> tensorrank(p1),tensorrank(p2);
```

4, 4

```
> r3:=gradedprod(p1,p2,B2,B1,B,B2);
```

$$\begin{aligned} r3 := & 2 \pi \cos(\alpha) (((e1 \& t e12) \& t e12) \& t e1) + \frac{6 \pi \cos(\alpha) a (((e1 \& t e1) \& t e12) \& t e1)}{a+c} \\ & - \frac{6 \pi \cos(\alpha) b (((e1 \& t e1) \& t e12) \& t e1)}{a+c} - 2 \pi \cos(\alpha) (((e1 \& t e2) \& t e12) \& t e1) \\ & - \frac{6 \pi \cos(\alpha) a (((e1 \& t Id) \& t e12) \& t e1)}{a+c} + \frac{6 \pi \cos(\alpha) b (((e1 \& t Id) \& t e12) \& t e1)}{a+c} \end{aligned}$$

```
> gcollect(r3);
```

$$\begin{aligned} & -6 \frac{\pi \cos(\alpha) (a-b) (((e1 \& t Id) \& t e12) \& t e1)}{a+c} + \frac{6 \pi \cos(\alpha) (a-b) (((e1 \& t e1) \& t e12) \& t e1)}{a+c} \\ & + 2 \pi \cos(\alpha) (((e1 \& t e12) \& t e12) \& t e1) - 2 \pi \cos(\alpha) (((e1 \& t e2) \& t e12) \& t e1) \end{aligned}$$

Thus, element r3 belongs to an algebra  $\mathbb{Z}_2$ -isomorphic with  $Cl(B2,B1,B,B2)$ , that is,  $Cl(B3)$  where B3 is a quadratic form of signature (9,7).

```
>
```

```
>
```

**See Also:** [GTP:-`type/gradedmonom`](#), [GTP:-gbasis](#), [GTP:-`type/gradedodd`](#), [GTP:-grade](#), [GTP:-`&t`](#), [Clifford:-`type/tensorprod`](#), [GTP:-`type/tensorrank`](#), [GTP:-`type/gradedeven`](#)



**- Function:** GTP:-tensorrang - find the rank of a tensor or an elements of the type 'gradedpolynom'

**Calling Sequence:**

tensorrang(p);

**Parameters:**

p - graded polynomials of type [GTP:-`type/gradedpolynom`](#)

**- Description:**

- Procedure 'tensorrang' finds the rank of a graded polynomial, that is, an element of the graded tensor product  $Cl(B1) \&t Cl(B2) \&t \dots \&t Cl(Br)$  of some Clifford algebras  $Cl(B1), Cl(B2), \dots, Cl(Br)$ .
- This procedure is needed to determine if the tensors entered in [GTP:-gprod](#) or [GTP:-gradedprod](#) are of the same rank.
- If, by mistake, tensors of different ranks are detected in p, an error message is returned.

**- Examples:**

```
[ > restart:with(Clipford):with(GTP):
[ > tensorrang(e1 &t e2);
[
[                                     2
[ > tensorrang(&t((e1 + e2),e2,Id,e1+e2,Id));
[
[ Cliplus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type
[ ?cliprod for help.
[
[                                     5
[ > p:=a*(Id &t Id &t e2)-3*(e1we2 &t e2 &t e2we1 )+b*4*(Id &t e2 &t e1)-
[     Id &t e1we2 &t e1we2;
[     p:=a((Id &t Id) &t e2) - 3((e1we2 &t e2) &t e2we1) + 4 b ((Id &t e2) &t e1) - ((Id &t e1we2) &t e1we2)
[ > tensorrang(p);
[
[                                     3
[ > p:=&t(e1,e2,e3)-&t(e1,e2);
[
[                                     p := ((e1 &t e2) &t e3) - (e1 &t e2)
[ > tensorrang(p); #testing an error message
[ Error, (in GTP:-tensorrang) tensors of mixed ranks are found in &t(&t(e1,e2),e3)-&t(e1,e2)
[ >
[ >
```

**- See Also:** [GTP:-`type/gradedmonom`](#), [GTP:-gbasis](#), [GTP:-`type/gradedodd`](#), [GTP:-grade](#), [GTP:-`&t`](#), [Clifford:-`type/tensorprod`](#), [GTP:-gradedprod](#), [GTP:-`type/gradedeven`](#)

Last revised: November 5, 2002/RA