DATA STRUCTURE fullmatrix

```
struct _fullmatrix {
    int  rows;
    int  cols;
    double  *e;
};
```

DATA STRUCTURE fullmatrix

```
typedef struct _fullmatrix fullmatrix;
typedef fullmatrix * pfullmatrix;
```

```
struct _fullmatrix {
    int  rows;
    int  cols;
    double  *e;
};
```

DATA STRUCTURE `fullmatrix`

```
typedef struct _fullmatrix fullmatrix;
typedef fullmatrix * pfullmatrix;
```

```
struct _fullmatrix {
    int  rows;
    int  cols;
    double  *e;
};
```

- The `double`-array `*e` is not allocated
- The values `rows`,`cols` are not initialised

Create and Delete object `fullmatrix`

```
struct _fullmatrix{
      int  rows;
      int  cols;
      double  *e;
};
```

CREATE AND DELETE OBJECT fullmatrix

```
struct _fullmatrix{
      int  rows;
      int  cols;
      double  *e;
};
```

```
pfullmatrix new_fullmatrix( int rows_new, int cols_new ){
      pfullmatrix f;
      f = (pfullmatrix) malloc(sizeof(fullmatrix));
      assert(f != 0x0);
      f->rows = rows_new;
      f->cols = cols_new;
      f->e = allocate_matrix( rows_new, cols_new );
      return f;
}
```

CREATE AND DELETE OBJECT `fullmatrix`

```
struct _fullmatrix{
      int  rows;
      int  cols;
      double  *e;
};
```

```
void  del_fullmatrix(pfullmatrix  f){
      assert(f!=0x0);
      assert(f->e!=0x0);
      freemem(f->e);
      free(f);
}
```

USAGE IN A PROGRAM

TASK: Create a `rows`$\times$`cols` dense matrix `f` with entries from the integral-equation example:

$$\mathtt{f}_{ij} := G_{\mathtt{rstart}+\mathtt{i},\mathtt{cstart}+\mathtt{j}}, \quad G \in \mathbb{R}^{\mathtt{n}\times\mathtt{n}}$$

USAGE IN A PROGRAM

TASK: Create a `rows`$\times$`cols` dense matrix `f` with entries from the integral-equation example:

$$f_{ij} := G_{\texttt{rstart+i},\texttt{cstart+j}}, \quad G \in \mathbb{R}^{\texttt{n}\times\texttt{n}}$$

```
pfullmatrix build_1d_fullmatrix(int n, int rows, int cols,
                                int rstart, int cstart){
    int i,j;
    pfullmatrix f;
    f = new_fullmatrix(rows,cols);



    return f;
}
```

The function `get_1d_entry` provides the entries of G

USAGE IN A PROGRAM

TASK: Create a `rows`$\times$`cols` dense matrix `f` with entries from the integral-equation example:

$$f_{ij} := G_{\texttt{rstart+i},\texttt{cstart+j}}, \quad G \in \mathbb{R}^{\texttt{n}\times\texttt{n}}$$

```
pfullmatrix build_1d_fullmatrix(int n, int rows, int cols,
                                int rstart, int cstart){
    int i,j;
    pfullmatrix f;
    f = new_fullmatrix(rows,cols);
     for(i=0; i<rows; i++)
        for(j=0; j<cols; j++)
           f->e[i+j*rows] = get_1d_entry(n,i+rstart,j+cstart);
     return f;
}
```

The function `get_1d_entry` provides the entries of G

DATA-STRUCTURE rkmatrix

```
struct _rkmatrix {
     int  rows;
     int  cols;
     int  k;
     int  kt;
     double  *a;
     double  *b;
};
```

DATA-STRUCTURE rkmatrix

```
typedef struct _rkmatrix rkmatrix;
typedef rkmatrix * prkmatrix;
```

```
struct _rkmatrix {
    int  rows;
    int  cols;
    int  k;
    int  kt;
    double  *a;
    double  *b;
};
```

DATA-STRUCTURE rkmatrix

```
typedef struct _rkmatrix rkmatrix;
typedef rkmatrix * prkmatrix;
```

```
struct _rkmatrix {
    int  rows;
    int  cols;
    int  k;
    int  kt;
    double  *a;
    double  *b;
};
```

- k: maximal rank (allocated storage)
- kt: actual rank (used rank)
- The double-arrays *a, *b are not allocated
- The values rows,cols,k,kt are not initialised

CREATE AND DELETE OBJECT rkmatrix

```
struct _rkmatrix {
      int  rows, cols;
      int  k;
      int  kt;
      double  *a, *b;
};
```

CREATE AND DELETE OBJECT rkmatrix

```
struct _rkmatrix{
      int  rows, cols;
      int  k;
      int  kt;
      double  *a, *b;
};
```

```
prkmatrix new_rkmatrix( int k_new, int rows_new, int cols_new){
      prmatrix r;
      r = (prkmatrix) malloc(sizeof(rkmatrix));
      assert(r != 0x0);
      r->k = k_new;
      r->kt = 0;
      r->rows = rows_new;  r->cols = cols_new;
      r->a = 0x0;  r->b = 0x0;
      if(k_new>0){
            r->a = allocate_matrix( rows_new, k_new);
            r->b = allocate_matrix( cols_new, k_new);
      }
      return r;
}
```

CREATE AND DELETE OBJECT rkmatrix

```
struct _rkmatrix {
      int  rows, cols;
      int  k;
      int  kt;
      double  *a, *b;
};
```

```
void del_rkmatrix(prkmatrix r){
      assert(r!=0x0);
      if(r->k>0){
            assert(r->a!=0x0);
            freemem(r->a);
            assert(r->b!=0x0);
            freemem(r->b);
      }
      free(r);
}
```

USAGE IN A PROGRAM

TASK: Create a `rows`$\times$`cols` `rkmatrix` `r` with entries from the integral-equation example:

$$r_{ij} := \tilde{G}_{\texttt{rstart+i},\texttt{cstart+j}}, \quad G \in \mathbb{R}^{\texttt{n}\times\texttt{n}}$$

USAGE IN A PROGRAM

TASK: Create a `rows`$\times$`cols` `rkmatrix` `r` with entries from the integral-equation example:

$$r_{ij} := \tilde{G}_{\texttt{rstart+i},\texttt{cstart+j}}, \quad G \in \mathbb{R}^{\texttt{n}\times\texttt{n}}$$

```
prkmatrix build_1d_rkmatrix(int n, int k, int rows, int cols,
                            int rstart, int cstart){
    int i,j,l;
    double x0;
    prkmatrix r;
    r = new_rkmatrix(k,rows,cols);









    return r;
}
```

The function `get_1d_taylorentry_a/b` yields the entries of A, B

USAGE IN A PROGRAM

TASK: Create a `rows`$\times$`cols` `rkmatrix` `r` with entries from the integral-equation example:

$$r_{ij} := \tilde{G}_{\texttt{rstart+i},\texttt{cstart+j}}, \quad G \in \mathbb{R}^{\texttt{n}\times\texttt{n}}$$

```
prkmatrix build_1d_rkmatrix(int n, int k, int rows, int cols,
                            int rstart, int cstart){
  int i,j,l;
  double x0;
  prkmatrix r;
  r = new_rkmatrix(k,rows,cols);
  r->kt = k;









  return r;
}
```

The function `get_1d_taylorentry_a/b` yields the entries of A, B

USAGE IN A PROGRAM

TASK: Create a `rows`$\times$`cols` `rkmatrix` `r` with entries from the integral-equation example:

$$r_{ij} := \tilde{G}_{\texttt{rstart+i},\texttt{cstart+j}}, \quad G \in \mathbb{R}^{\texttt{n}\times\texttt{n}}$$

```
prkmatrix build_1d_rkmatrix(int n, int k, int rows, int cols,
                            int rstart, int cstart){
  int i,j,l;
  double x0;
  prkmatrix r;
  r = new_rkmatrix(k,rows,cols);
  r->kt = k;
  x0 = ((double)(2*rstart+rows))/((double)2.0*n);









  return r;
}
```

The function `get_1d_taylorentry_a/b` yields the entries of A, B

USAGE IN A PROGRAM

TASK: Create a `rows`$\times$`cols` `rkmatrix` `r` with entries from the integral-equation example:

$$r_{ij} := \tilde{G}_{\texttt{rstart+i},\texttt{cstart+j}}, \quad G \in \mathbb{R}^{\texttt{n}\times\texttt{n}}$$

```
prkmatrix build_1d_rkmatrix(int n, int k, int rows, int cols,
                            int rstart, int cstart){
   int i,j,l;
   double x0;
   prkmatrix r;
   r = new_rkmatrix(k,rows,cols);
   r->kt = k;
   x0 = ((double)(2*rstart+rows))/((double)2.0*n);
   for(l=0; l<k; l++)
      for(i=0; i<rows; i++)
         r->a[i+l*rows] = get_1d_taylorentry_a(n,i+rstart,l,x0);



    return r;
}
```

The function `get_1d_taylorentry_a/b` yields the entries of A, B

USAGE IN A PROGRAM

TASK: Create a `rows`×`cols` `rkmatrix` `r` with entries from the integral-equation example:

$$r_{ij} := \tilde{G}_{\texttt{rstart+i},\texttt{cstart+j}}, \quad G \in \mathbb{R}^{\texttt{n}\times\texttt{n}}$$

```
prkmatrix build_1d_rkmatrix(int n, int k, int rows, int cols,
                            int rstart, int cstart){
    int i,j,l;
    double x0;
    prkmatrix r;
    r = new_rkmatrix(k,rows,cols);
    r->kt = k;
    x0 = ((double)(2*rstart+rows))/((double)2.0*n);
    for(l=0; l<k; l++)
       for(i=0; i<rows; i++)
          r->a[i+l*rows] = get_1d_taylorentry_a(n,i+rstart,l,x0);
    for(l=0; l<k; l++)
       for(j=0; j<cols; j++)
          r->b[j+l*cols] = get_1d_taylorentry_b(n,j+cstart,l,x0);
     return r;
}
```

The function `get_1d_taylorentry_a/b` yields the entries of A, B

DATA STRUCTURE supermatrix

```
struct _supermatrix {
    int  rows;
    int  cols;
    int  block_rows;
    int  block_cols;
    psupermatrix *s;
    prkmatrix r;
    pfullmatrix f;
};
```

DATA STRUCTURE supermatrix

```
typedef struct _supermatrix supermatrix;
typedef supermatrix * psupermatrix;
```

```
struct _supermatrix {
    int  rows;
    int  cols;
    int  block_rows;
    int  block_cols;
    psupermatrix *s;
    prkmatrix r;
    pfullmatrix f;
};
```

DATA STRUCTURE supermatrix

```
typedef struct _supermatrix supermatrix;
typedef supermatrix * psupermatrix;
```

```
struct _supermatrix {
    int  rows;
    int  cols;
    int  block_rows;
    int  block_cols;
    psupermatrix *s;
    prkmatrix r;
    pfullmatrix f;
};
```

- Subdivision into `block_rows` × `block_cols` submatrices
- `s[i+j*block_rows]`: (i,j)-th submatrix
- If not subdivided then either
  - `rkmatrix`: `r`, or
  - `fullmatrix`: `f`