図 4-6 のデータ構造を用いて表された実数値行列 *A, B* の積 *C* を求めるプログラム

```
typedef struct {
        CELL↑    row_header[1..maxlength] ;
        CELL↑    col_header[1..maxlength] ;
} Sparse_Matrix ;

typedef struct {
        int    row, col ;
        float  element ;
        CELL↑    r_next ;
        CELL↑    c_next ;}
} CELL ;
```

```
void   Sparse_Matrix_Product(  n, A, B, C  )
    入力:    n  :    整数 (値呼び)
             A,  B  :  Sparse_Matrix
    出力:    C  :    Sparse_Matrix
/*  図 4.6 のような Sparse_Matrix で与えられた行列 A, B, の積 C を計算する手続き  */
{
    int          i,  j  ;                    /*  行 i, 列 j を指定するための変数  */
    float        sum  ;                      /*  行列 C の i, j 要素を計算するために使う  */
    CELL↑    r,  c  ;                        /*  セルを調べるために使うポインタ  */
    CELL↑    r_last  ;                 /*  処理中の行 i の一番後ろのセルを指すポインタ  */
    CELL↑    ptr  ;                      /*  新たに生成したセルを指すためのポインタ  */
    CELL↑    col_last[1..n]  ;
                 /*  col_last[j] は行列 C の列 j の一番下のセルを指すポインタ  */
                 /*  新たな行 i の処理において, 列 j に非零要素が生じたとき  */
                 /*  col_last[j] の指すセルの下にこの非零要素のセルを生成する  */
    /*  col_last[・] と C.col_header[・] の初期設定処理  */
    for (  1≦j≦n なる各 j  ) {
        col_last[j]  :=  NULL  ;
        C.col_header[j]  :=  NULL  ;
    }
    /*  行列 C[・] の i, j 要素を計算するための処理  */
```

```
    for ( 1≦i≦n なる各 i ) {                    /* 各行 i 毎の処理 */
        C.row_header[i] := NULL ;                /* C.row_header[・] の初期設定 */
        r_last := NULL ;
                    /* r_last は行列 C の行 i の一番後ろのセルを指すポインタ */
        for ( 1≦j≦n なる各 j ) {                /* 各行 j 毎の処理 */
            r := A.row_header[i] ;               /* 行列 A の行 i のセルを指すのに使う */
            c := B.col_header[j] ;               /* 行列 B の列 j のセルを指すのに使う */
            sum := 0 ;

            while ( r≠NULL かつ c≠NULL ) {
                if ( (↑r).col < (↑c).row )       r := (↑r).r_next ;
                else if ( (↑r).col > (↑c).row )  c := (↑c).c_next ;
                else {          /* A[i,k], B[k,j] 共に非零. k = (↑r).col = (↑c).row */
                    sum := sum + (↑r).element×(↑c).element ;
                    r := (↑r).r_next ;
                    c := (↑c).c_next ;
            }   }
            if ( sum≠0 ) {                       /* 行列 C の i, j 要素は非零 */
                ptr := new( CELL ) ;             /* ptr は新たに生成したセルを指す */
                if ( r_last = NULL )             C.row_header[i] := ptr ;
                else                             (↑r_last).r_next := ptr ;
                r_last := ptr ;
                if ( col_last[j] = NULL )        C.col_header[j] := ptr ;
                else                             col_last[j].c_next := ptr ;
                col_last[j] := ptr ;
                (↑ptr).row := i ;
                (↑ptr).col := j ;
                (↑ptr).element := sum ;
                (↑ptr).r_next := NULL ;
                (↑ptr).c_next := NULL ;
    }   }   }
} /* Sparse_Matrix_Product */
```

本の 2 ページのプログラムに比べて, 上記は長いプログラムであるが, 文の多さは計算時間に直接影響しない. 計算時間は終了までに何個の文を実行するかで決まる.