```
typedef struct {
	long dgrlhsize;
	char dgrlcode[8];
	char* illustr;
	char codetype[20];
	short codelen;
	short bitspp;
} SDGRLHeader;

CFile imageFile;   // File of character images
BYTE fileOpen;          // The file is open (1) or not (0)
CString sFTitle;

// Header of MGI file
SDGRLHeader dgrlh;
long pageHei, pageWid; //size of page image
long lineNumber;

long charNumber[100];// number of images per line
char* lineLabel[100];
BYTE* lineImage[100];      // at most 100 lines on a page

long lineLeft[100], lineTop[100]; // top-left position of line image
long lineHei[100], lineWid[100];

void readImage();  // Read a page image

CString sFName;

void DGRLRead()
{
	CFileException fileException;
	imageFile.Open(sFName, CFile::modeReadWrite, &fileException);
	fileOpen = 1;

	imageFile.Read((&dgrlh.dgrlhsize), 4);
	imageFile.Read((&dgrlh.dgrlcode), 8);

	int illuslen = dgrlh.dgrlhsize - 36;
	dgrlh.illustr = new char[illuslen];
	imageFile.Read(dgrlh.illustr, illuslen);

	imageFile.Read(dgrlh.codetype, 20);
	imageFile.Read(&(dgrlh.codelen), 2);
	imageFile.Read(&(dgrlh.bitspp), 2);

	readImage();

	for (long n = 0; n < lineNumber; n++)
	{
```

```
            delete lineLabel[n];
            delete lineImage[n];
        }
        delete dgrlh.illustr;
}

void readImage()
{
        if (fileOpen == 0)
            return;

        imageFile.Read(&pageHei, 4);
        imageFile.Read(&pageWid, 4);
        imageFile.Read(&lineNumber, 4);

        for (long n = 0; n < lineNumber; n++)
        {
            imageFile.Read(charNumber + n, 4);
            lineLabel[n] = new char[charNumber[n] * dgrlh.codelen+1];

            imageFile.Read(lineLabel[n], charNumber[n] * dgrlh.codelen);
            lineLabel[n][charNumber[n] * dgrlh.codelen] = 0;
            for (int i = 0; i < charNumber[n] * dgrlh.codelen; i++)
            {
                if (lineLabel[n][i] == 0)
                    lineLabel[n][i] = ' ';
                // for single-byte characters (such as alphabet), replace the second byte with '-', for
convenient of display
            }

            imageFile.Read(lineTop + n, 4);
            imageFile.Read(lineLeft + n, 4);
            imageFile.Read(lineHei + n, 4);
            imageFile.Read(lineWid + n, 4);

            lineImage[n] = new BYTE[lineHei[n] * lineWid[n]];
            imageFile.Read(lineImage[n], lineHei[n] * lineWid[n]);
        }
}

/*
Care should be taken when concatenating the lines into page image. Different lines may have
overlap of plane, because
the text strokes of different lines may overlap in vertical axis. So, for restoring the page image,
the foreground pixels of
different lines should be combined.
*/
```

Format of DGRL file

A DGRL (*.dgrl) file stores a page of document image. The image has background eliminated (encoded as 255) and foreground (text strokes) encoded in gray level 0-254, one byte per pixel. Each page is stored as a series of lines. Each line has a header denoting the number of characters, sequence of character codes (GBK), top-left position, line height and width, then the block of bitmap (height*width bytes).

For concatenating the lines into page image, it should be noted that different lines may have overlap of plane, because the text strokes of different lines may overlap in vertical axis. So, for restoring the page image, the foreground pixels of different lines should be ORed.

Table 4. Format of offline text data file (*.dgrl)

Item	Type	Length	Instance
File Header			
Size of Header	int	4B	Number of bytes: 36+strlen(illustr)
Format code	ASCII (char*)	8B	"DGRL"
Illustration	Text	Arbitrary	"#.....\0"
Code type	ASCII (char*)	20B	"ASCII", "GB", etc.
Code length	Short	2B	1, 2, 4, etc.
Bits per pixel	Short	2B	Typically 1(B/W image), 8 (Gray image)
Image Records (concatenated)			
Image height	int	4B	Height (pixels) of document image
Image width	int	4B	Width (pixels) of document image
Line number	int	4B	Number of lines in the image
Line Records (concatenated)			
Char number	int	4B	Number of characters in a line
Label (code)	Code type	Code length* Char number	Each byte is 0xff(-1) for garbage
Top-left coordinates	int	4B + 4B	(top, left) of a line
Height (H)	int	4B	Height (pixels) of a line
Width (W)	int	4B	Width (pixels) of a line
Bitmap	BYTE	H*((W + 7) / 8) or H*W	Binary or gray image