オブジェクト指向とゲームプログラミング

● 基礎編 ― 第23回　ファイルの操作③ ●

ファイルの形式

ファイルの扱い方にはテキストモードとバイナリモードの２種類あります。テキストモードでファイルをオープンすると、Windowsの改行コード(Ｃ言語では'\n')である「0x0D」「0x0A」の２バイトが、「0x0A」の１バイトに変換されます。これは、UNIX系のOSと互換をとるためです。また、ファイル中の「0x1A」という値は、ファイルの終端(EOF:End Of File)として認識され、継続するデータがあったとしても、以降のデータはいっさい読み込むことができません。

これに対してバイナリモードでファイルをオープンすると、いっさいの変換は行われません。ファイルのデータがそのまま読み込まれ、書き込んだデータはそのままファイルに記録されます。「0x1A」もEOFとして扱われず、そのまま読み込むことができ、以降のデータももちろん読み込めます。

テキストファイルなど文字列を扱うファイルはテキストモード、ビットマップファイルや音声ファイルなど、データを扱うファイルはバイナリモードでオープンします。

バイナリモードのファイルの読み込み

バイナリモードでは、以下の標準関数を用いてファイルの読み書きを行います。

fread関数

fread関数は、ファイルから指定された領域に、指定されたバイト数のデータを指定された個数読み込み、読み込んだデータのバイト数ファイルポインタを進めます。戻り値はデータを読み込んだ個数が返ります。エラーの場合は「指定された個数」より小さい値となります。

```
FILE*   fp = fopen("Data\\Chara00.dat", "rb");      // ファイルのオープン

int     in_data1[2];                                // 入力領域その１
char    in_data2[10];                               // 入力領域その２
long    in_data3;                                   // 入力領域その３

// ファイルからint型のデータを２個読み込む
fread(in_data1,  sizeof(int),   2, fp);

// ファイルからchar型のデータを１０個読み込む
fread(in_data2,  sizeof(char), 10, fp);

// ファイルからlong型のデータを１個読み込む
fread(&in_data3, sizeof(long),  1, fp);
```

fwrite関数

fwrite関数は、指定された領域からファイルに、指定されたバイト数のデータを指定された個数書き込み、書き込んだデータのバイト数ファイルポインタを進めます。戻り値はデータを書き込んだ個数が返ります。エラーの場合は「指定された個数」より小さい値となります。

```
FILE*   fp = fopen("Save\\Save00.sav", "wb");       // ファイルのオープン

int     out_data1[2];                               // 入力領域その１
char    out_data2[10];                              // 入力領域その２
long    out_data3;                                  // 入力領域その３

// ファイルにint型のデータを２個書き込む
fwrite(out_data1,  sizeof(int),   2, fp);

// ファイルにchar型のデータを１０個書き込む
fwrite(out_data2,  sizeof(char), 10, fp);

// ファイルにlong型のデータを１個書き込む
```

```
fwrite(&out_data3, sizeof(long),  1, fp);
```

fseek関数

fseek関数は、ファイルポインタの位置を変更します。この関数の書式は以下のようになっています。

```
fseek(FILE* stream, long offset, int origin)
```

ファイルポインタstreamが指すファイルの読み書き位置を、originからoffsetバイト増減します。originには、以下の定数を指定します。

SEEK_CUR	ファイルポインタの現在位置
SEEK_SET	ファイルの先頭
SEEK_END	ファイルの終端

関数が正常に処理された場合は０が返ります。エラーの場合は０以外の値が返ります。

```
// ファイルポインタを先頭に戻す
fseek(fp, 0, SEEK_SET);

// ファイルポインタを終端にする
fseek(fp, 0, SEEK_END);

// ファイルポインタを現在の位置から２５６バイト進める
fseek(fp, 256, SEEK_CUR);
```

課　題

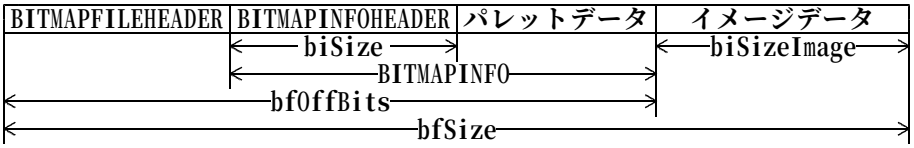
以下の仕様を満たすコンソールアプリケーションを作成しましょう。

－仕様－
・ビットマップファイルから、画像の幅、高さ、色数、イメージのバイト数を取得し、表示する
・パレットがある場合は、すべてのパレットを赤、緑、青の成分ごとに表示する
(例：「No. 0 : R: 0 G: 96 b:255」)
・コマンドラインから、情報を取得するビットマップのファイル名を指定することができる
・ファイル名が指定されずに起動した場合、「ファイルを開く」ダイアログを表示する
・ビットマップファイルでない場合は、「ビットマップファイルではありません」と表示

ビットマップファイル

ビットマップファイルは、Windowsが標準でサポートしている画像フォーマットです。拡張子は「bmp」です。ビットマップは、白黒(２色)から16色、256色、フルカラー(16,777,216色)までの色数をサポートしています。基本的にイメージをそのまま保存するので、ファイルサイズが大きくなってしまいます。RLE(ランレングス)という方法で圧縮することもできますが、正しく読み込めない場合もあります。

ファイルに保存されたビットマップは、以下のような形式になります。

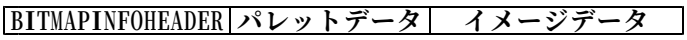


ビットマップファイルの先頭には、BITMAPFILEHEADER構造体があります。この構造体には、ファイルの種類、サイズ、データの配置に関する情報が格納されています。先頭のメンバ(bfType)に、ファイルの種類が格納されています。ビットマップファイルの場合は、ここに'BM'の２文字(※注：文字列ではないので'\0'はありません)が格納されます。よって、ファイルの先頭２バイトが'BM'であるかどうかを調べれば、ビットマップファイルかどうかを調べることができます。bfSizeメンバにはファイルサイズ、bfOffBitsメンバにはイメージデータまでのバイト数(オフセット値)が格納されています。

BITMAPFILEHEADER構造体の次に格納されているのがBITMAPINFOHEADER構造体です。この構造体には、カラー形式や寸法などが格納されています。特に重要なメンバがbiWidth, biHeight, biBitCountメンバです。
biWidth, biHeightメンバにはイメージの幅と高さが格納されています。biBitCountメンバには色数が格納されています。この値が１(モノクロ)、４(16色)、８(256色)の場合には、パレットデータを保有し、イメージデータはそのパレットデータのインデックス値となることを表しています。たとえば、４ビットカラービットマップでは、最大16個のパレットがあり、イメージデータは１ピクセルあたり４ビット(0～15の値)で構成されます。
16, 24, 32ビットのビットマップでは、イメージデータがそのままRGB値を表現しています。24ビットカラーのビットマップは、青、緑、赤の順でそれぞれ１バイト、計３バイトで１つのピクセルを表現します。

BITMAPINFOHEADER構造体の後にはRGBQUAD構造体型のパレットデータが続きます。パレットが存在する場合、BITMAPINFOHEADER構造体のbiClrUsedメンバの値がパレットの数を表していますが、この値に０が設定されている場合もあります。その場合には、biBitCountメンバでピクセルあたりのビット数により最大カラー数を求めて、それをパレットデータの数とします。

Windowsでは、ファイルのほかにリソースにもビットマップを保存することができます。リソースに保存した場合は、以下のように、ビットマップファイルからBITMAPFILEHEADER構造体を取り除いた形式になります。



・BITMAPFILEHEADER構造体
ファイルの種類、サイズ、データの配置に関する情報を保持します。
－メンバー－
WORD　bfType……'BM'の文字コード
DWORD　bfSize……ファイルサイズ
WORD　bfReserved1 … 予約されているので、０になります
WORD　bfReserved2 … 予約されているので、０になります
DWORD　bfOffBits… …イメージデータまでのバイト数

・BITMAPINFO構造体
ビットマップの寸法と色の情報を保持します。
－メンバー－
BITMAPINFOHEADER bmiHeader ……次項を参照してください
RGBQUAD　bmiColors[1]…色を定義するRGBQUAD構造体型の配列

・BITMAPINFOHEADER構造体
ビットマップの寸法とカラーフォーマットに関する情報を保持します。
－メンバー－
DWORD　biSize……構造体のサイズ
LONG　biWidth……ビットマップの幅
LONG　biHeight……ビットマップの高さ
WORD　biPlanes……プレーンの数(ビットマップ形式では必ず１)
WORD　biBitCount… … …ピクセルあたりのビット数
DWORD　biCompression……圧縮形式
DWORD　biSizeImage……イメージのサイズ(バイト)
LONG　biXPelsPerMeter…ターゲットデバイスの水平解像度(ピクセル数／m)
LONG　biYPelsPerMeter…ターゲットデバイスの垂直解像度(ピクセル数／m)
DWORD　biClrUsed……パレットの数
DWORD　biClrImportant …表示において重要なパレットの数

－備考－
biClrUsedが０…biBitCountメンバの値をもとに、最大の色数を使用
biClrUsedが０以外かつbiBitCountが16未満…biClrUsedは実際の色数を示します
biClrUsedが０以外かつbiBitCoundが16以上…Windowsのパレットデーブルのサイズを示します
biClrImportantが０…すべての色が重要

・RGBQUAD構造体
赤、緑、青の相対的な輝度で構成される色を表します。
－メンバー－
BYTE　rgbBlue……青の輝度
BYTE　rgbGreen ……緑の輝度
BYTE　rgbRed ……赤の輝度
BYTE　rgbReserved…予約されています。0でなければなりません

●２５６色ビットマップファイルの構造

－画像(320×240)－

ビットマップファイルヘッダ
 	
BITMAPFILEHEADER
		struct BITMAPFILEHEADER {
		WORD bfType = 'BM'(0x424D)
		DWORD bfSize = 77,878
		WORD bfReserved1 = 0
		WORD bfReserved2 = 0
		DWORD bfOffBits = 1,078
		};

ビットマップインフォヘッダ
 	
BITMAPINFOHEADER
		struct BITMAPINFOHEADER{
		DWORD biSize = 40
		LONG biWidth = 320
		LONG biHeight = 240
		WORD biPlanes = 1
		WORD biBitCount = 8
		DWORD biCompression = BI_RGB
		DWORD biSizeImage = 0または76,800
		LONG biXPelsPerMeter = 0
		LONG biYPelsPerMeter = 0
		DWORD biClrUsed = 0または256
		DWORD biClrImportant = 0
		};

カラーテーブル
 	
RGBQUAD×色数
		struct RGBQUAD {
		BYTE rgbBlue;
		BYTE rgbGreen;
		BYTE rgbRed;
		BYTE rgbReserved;
		};
		 

イメージデータ ※上下が反転しています ※１ラインは４の倍数に合わせられるため、余分なデータが付加される場合があります。 ※256色以下の場合、右のような色値そのものではなく、色値に対応するパレットの番号が保存されます