オブジェクト指向と ゲームプログラミング

● 基礎編 ― 第24回 ファイルの操作④ ●

APIによるファイルの操作

APIにもファイルを扱うための関数がいくつか用意されています。APIを使った場合、ファイルの操作は、以下の流れで行います。

①CreateFile関数でファイルを開く
②ReadFile関数やWriteFile関数でファイルへ入出力
③CloseHandle関数でファイルを閉じる

CreateFile関数
―説明―
CreateFile関数は、ファイルや通信リソースなどを作成またはオープンし、それらにアクセスするために利用できるハンドルを返します。

―パラメータ―
１つ目の引数は、作成またはオープンするファイル名を指定します。

２つ目の引数は、アクセスの種類を指定します。次の値を組み合わせて指定します。

0	デバイスの種類を問い合わせます
GENERIC_READ	読み取りアクセス
GENERIC_WRITE	書き込みアクセス

３つ目の引数は、ファイルの共有方法を指定します。ファイルを共有するには、次の値を組み合わせて指定します。

0	共有はされません。後続のオープン作業は、ファイルをクローズしない限り失敗します。
FILE_SHARE_READ	後続のオープン操作で読み取りアクセスが要求された場合、そのオープンを許可します。
FILE_SHARE_WRITE	後続のオープン操作で書き込みアクセスが要求された場合、そのオープンを許可します。

４つ目の引数は、ファイルに対するセキュリティ属性を指定する構造体のアドレスです。通常はNULLを指定します。

５つ目の引数は、ファイルが存在するとき、または存在しないときのファイルの動作を指定します。次の値のいずれかを指定します。

CREATE_NEW	新しいファイルを作成します。指定ファイルがすでに存在している場合、関数は失敗します。
CREATE_ALWAYS	新しいファイルを作成します。指定ファイルがすでに存在している場合、そのファイルは上書きされます。
OPEN_EXISTING	ファイルをオープンします。指定ファイルが存在していない場合、関数は失敗します。
OPEN_ALWAYS	ファイルをオープンします。指定ファイルが存在していない場合、新しいファイルを作成します。
TRUNCATE_EXISTING	ファイルをオープンし、ファイルのサイズを０バイトにします。指定ファイルが存在していない場合、関数は失敗します。ファイルモードでGENERIC_WRITEを指定しなければなりません。

６つ目の引数は、ファイルの属性およびフラグを指定します。読み取り専用や隠しファイルなどを指定することができます。通常はFILE_ATTRIBUTE_NORMALを指定します。

７つ目の引数は、コピーする属性付きファイルのハンドルです。通常は使用しないので、NULLを指定します。

―戻り値―
成功した場合は指定したファイルに対するハンドルが返ります。失敗した場合はINVALID_HANDLE_VALUEが返ります。

```
// 読み込み専用アクセス
HANDLE   hReadFile  = CreateFile("D:\\Dat\\Data.dat", GENERIC_READ, FILE_SHARE_READ,
                                  NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
// 書き込み専用アクセス
HANDLE   hWriteFile = CreateFile("D:\\Dat\\Data.dat", GENERIC_WRITE, 0,
                                  NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);

// 読み書きアクセス
HANDLE   hFile = CreateFile("D:\\Dat\\Data.dat", GENERIC_READ | GENERIC_WRITE, 0,
                            NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
```

CloseHandle関数
－説明－
CloseHandle関数は、生成またはオープンされたハンドルをクローズします。

－パラメータ－
引数は、クローズするハンドルです。

－戻り値－
成功すると０以外の値が返ります。失敗すると０が返ります。

```
// ファイルクローズ
CloseHandle(hFile);
```

ReadFile関数
－説明－
ReadFile関数は、ファイルからデータを読み取ります。ファイルポインタ(この場合は、ファイルを読み書きする位置のことを指します)の現在の位置が、読み取りの開始位置になります。読み取りが完了すると、ファイルポインタの位置は、実際に読み取ったバイト数だけ進みます。

－パラメータ－
１つ目の引数は、読み取り対象のファイルのハンドルを指定します。このハンドルは、GENERIC_READアクセス権を指定して作成したものでなければなりません。

２つ目の引数は、読み込んだデータを格納する領域のアドレスを指定します。この領域にファイルから読み込んだデータが格納されます。

３つ目の引数は、読み込むバイト数を指定します。

４つ目の引数は、DWORD型の変数のアドレスを指定します。関数から制御が返ると、この変数に実際に読み込まれたバイト数が格納されます。

５つ目の引数は、オーバーラップ構造体のアドレスを指定します。通常は使用しないのでNULLを指定します。

－戻り値－
成功すると０以外の値が返り、失敗すると０が返ります。戻り値が０以外で４つ目の引数の内容が０になった場合、読み取り操作を開始した時点でファイルポインタが終端を超えていたことを示します。

```
DWORD   actual;     // 実際に読み込んだバイト数を受け取る変数

// BMPヘッダ読込み
BITMAPFILEHEADER   bmfh;
ReadFile(hReadFile, &bmfh, sizeof(bmfh), &actual, NULL);
```

WriteFile関数
－説明－
WriteFile関数は、ファイルにデータを書き込みます。ファイルポインタの現在の位置が、書き込みの開始位置になります。書き込みが完了すると、ファイルポインタの位置は、実際に書き込んだバイト数だけ進みます。

－パラメータ－
　１つ目の引数は、書き込み対象のファイルのハンドルを指定します。このハンドルは、GENERIC_WRITEアクセス権を指定して作成したものでなければなりません。

　２つ目の引数は、書き込むデータが格納されている領域のアドレスを指定します。
　３つ目の引数は、書き込むバイト数を指定します。

　４つ目の引数は、DWORD型の変数のアドレスを指定します。関数から制御が返ると、この変数に実際に書き込まれたバイト数が格納されます。

　５つ目の引数は、オーバーラップ構造体のアドレスを指定します。通常は使用しないのでNULLを指定します。

－戻り値－
　成功すると０以外の値が返ります。失敗すると０が返ります。

```
DWORD   actual;      // 実際に書き込んだバイト数を受け取る変数

// ファイルに「YOSHIKI」の７バイト書き込む
WriteFile(hWriteFile, "YOSHIKI", 7, &actual, NULL);
```

　これらの関数以外にも、以下のように便利な関数が多数提供されています。

SetFilePointer関数	ファイルポインタを移動します。
GetFileSize関数	ファイルのサイズを取得します。
GetLogicalDrives関数	利用可能なドライブを取得します。
GetDriveType関数	ドライブの種類を取得します。

　APIによるファイルの入出力は、バイナリファイルを扱う場合に使用します。テキストファイルのような文字列の入出力は、標準ライブラリを使った方が簡単です。

課　題

　第１１回で作成したプログラムのfopen, fread, fseek, fclose関数を、CreateFile, ReadFile, SetFilePointer, CloseHandle関数に変更しましょう。