DDB - Device Dependent Bitmap

장치 종속적인 비트맵 이미지.

GDI 에서는 DC 와 연결되는 Bitmap 을 DDB 라고 한다. GDI 에서 비트맵 출력을 위해선 DDB 가 필요하며 장치 (DC) 에 종속되는 비트맵으로 색상,크기 등의 이미지 정보가 DC 를 따라가며 DC 와 호환이 되어야만 정상적으로 그림이 출력 됩니다..

CreateCompatibleBitmap, CreateBitmap 등의 함수를 사용하여 만들어진 HBITMAP 객체들이 DDB 이며, GDI 를 통해서 화면 (DC) 에 출력된 이미지는 DDB 라고 보아도 됩니다.

DIB - Device Independent Bitmap

장치에 종속되지 않는 독립적인 비트맵 입니다.

대표적으로 BMP 파일이 있으며, 이 BMP 파일에는 그림에 대한 모든 정보 (사이즈, 색상, 팔레트 등) 가 들어있다. 많은 정보가 들어있어서 BMP 파일 자체만으로도 이미지를 표현할 수 있습니다. 바꾸어 말하면 BMP 파일 하나만으로 그림을 정상적으로 출력할 수 있는 정보를 가지고 있는 것 입니다

이미지를 파일로 저장하기 위해선 독립적인 DIB 로 저장이 되어야 하며, 이를 윈도우에서 GDI 를 통해 화면에 출력 하려면 DDB 로 변환을 해야 합니다.

DIB (BMP) 의 구조

<BMP 파일의 구조>

struct BITMAPFILEHEADER

비트맵 파일의 정보를 가지는 구조체 입니다.

```
typedef struct tagBITMAPFILEHEADER {
  WORD  bfType;
  DWORD bfSize;
  WORD  bfReserved1;
  WORD  bfReserved2;
  DWORD bfOffBits;
} BITMAPFILEHEADER;
```

bfType – BM 글자
bfSize – 파일의 전체 크기
bfReserved1 – 사용하지 않음 무조건 0
bfReserved2 – 사용하지 않음 무조건 0
bfOffBits – 이미지가 시작되는 Offset 위치

BMP 파일의 최 상단은 위와 같은 파일정보가 들어갑니다. 이는 DIB 의 정보가 아닌 BMP 파일 자체의 정보로 DIB 구조에서는 필요하지 않는 정보 입니다.

우리는 BMP 파일을 오픈 하여 가장 먼저 BITMAPFILEHEADER 를 읽어 들여야 합니다. 그리고 bfType 값을 비교하여 BM 값이 맞는지 확인 합니다. 가장 앞에 BM 이 없다면 이는 BMP 파일이 아닌 것으로 판단 할 수 있습니다.

struct BITMAPINFOHEADER

```
typedef struct tagBITMAPINFOHEADER {
  DWORD biSize;
  LONG  biWidth;
  LONG  biHeight;
  WORD  biPlanes;
  WORD  biBitCount;
  DWORD biCompression;
  DWORD biSizeImage;
  LONG  biXPelsPerMeter;
  LONG  biYPelsPerMeter;
  DWORD biClrUsed;
  DWORD biClrImportant;
} BITMAPINFOHEADER
```

biSize – 본 구조체의 크기
biWidth – 필셀 너비 (바이트 길이가 아닌 픽셀)
biHeight – 픽셀 높이
biPlanes – 무조건 1
biBitCount – 색상 깊이 (8 : 256, 16 : 65536, 24, 32)
biCompression – 압축 방식 BI_RGB(0) 비트맵은 압축을 하지 않는 것이 기본
biSizeImage – 이미지 크기 무압축의 경우 사용되지 않음
biXPelsPerMeter – 디바이스 미터당 픽셀 수 (무시)
bbYPersPerMeter – 디바이스 미터당 픽셀 수 (무시)
biClrUsed – 실제 사용 색상 수 0 이면 올칼라
biClrImportant – 주로 사용된 색상 수 0 이면 올칼라

BITMAPINFOHEADER 정보를 읽어서 BMP 파일의 정보를 확인 할 수 있습니다. 이 데이터부터 DIB 의 자료 입니다.

struct RGBQUAD

8bit 컬러의 경우 팔레트 데이터를 사용합니다. RGBQUAD 값이 펠레트 데이터이며, 256 칼라라면 RGBQUAD 데이터가 256개 존재합니다. 하지만 8bit 색상은 거의 사용하지 않으므로 생략 합니다.

픽셀 이미지 데이터

이제 이미지의 크기 만큼 픽셀 데이터가 들어 있습니다. DIB 의 특징은 다음과 같습니다.

1. 이미지가 뒤집혀서 들어있다.
2. 줄단위로 4Byte 정렬이 되어있다.

DIB (BMP) 는 이미지가 뒤집혀서 들어있습니다.

이미지가 뒤집혀져 있지만 뒤집혀진 이미지가 정상 DIB 구조 이므로, 뒤집혀진 채로 읽어서 DIB 출력을 해주면 그림이 출력되게 됩니다. DIB 를 DC 에 출력하는 함수는 StretchDIBits 함수 또는 SetDIBitsToDevice 를 사용하게 됩니다.

```
int StretchDIBits(
  _In_  HDC hdc,
  _In_  int XDest,
  _In_  int YDest,
  _In_  int nDestWidth,
  _In_  int nDestHeight,
  _In_  int XSrc,
  _In_  int YSrc,
  _In_  int nSrcWidth,
  _In_  int nSrcHeight,
  _In_  const VOID *lpBits,
  _In_  const BITMAPINFO *lpBitsInfo,
  _In_  UINT iUsage,
  _In_  DWORD dwRop
);
```

hdc – 목적지 DC
XDest – 출력대상 X 위치
YDest – 출력대상 Y 위치
nDestWidth – 목적지 너비
nDestHeight – 목적지 높이
XSrc – 출력소스 X 위치
YSrc – 출력소스 Y 위치
nSrcWidth – 출력소스 너비
nSrcHeight – 출력소스 높이
lpBits – 이미지가 위치한 포인터
lpBitsInfo – BITMAPINFO 구조체의 위치 / BITMAPINFO 는 BITMAPINFOHEADER + RGBQUAD 가 합쳐진 구조체
iUsage - DIB_RGB_COLORS
dwRop – 출력모드 SRCCOPY

SetDIBitsToDevice 함수는 위와 비슷하며 차이점은 늘리기 줄이기 출력이 되는지 안 되는지 이다. 단순 1:1 출력만을 하고자 한다면 SetDIBitsToDevice 를 사용하는 것이 더 빠르고 인자가 단순하다.