1. Readig bitmap image informations.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "bmp.h"


unsigned short ReadLE2(FILE *fp);
unsigned int ReadLE4(FILE *fp);

/*
 * Read bitmap file header
 */
BITMAPFILEHEADER *ReadBMFileHeader(FILE *fp)
{
    BITMAPFILEHEADER *header;
    char           filetype[3] = {'\0', '\0', '\0'};
    unsigned int   filesize;
    unsigned short reserved1;
    unsigned short reserved2;
    unsigned long  offset;

    /* File type (2 bytes) */
    fread(&filetype, 1, 2, fp);

    /* File size (4 bytes) */
    filesize = (unsigned int) ReadLE4(fp);

    /* Reserved 1 (2 bytes) */
    fread(&reserved1, 2, 1, fp);

    /* Reserved 2 (2 bytes) */
    fread(&reserved2, 2, 1, fp);

    /* Offset (4 bytes) */
    offset = (unsigned long) ReadLE4(fp);

    header = (BITMAPFILEHEADER *) malloc(sizeof(BITMAPFILEHEADER));
    strcpy(header->bfType, filetype);
    header->bfSize      = filesize;
    header->bfReserved1 = reserved1;
    header->bfReserved2 = reserved2;
```

```
    header->bfOffBits   = offset;

    return header;
}

/*
 * Returns size of information header
 */
int SizeOfInformationHeader(FILE *fp)
{
    int headersize;
    unsigned char buf[4];
    int i;

    fread(buf, 1, 4, fp);
    for (i = 3; i >= 0; i--) {
        headersize = (headersize << 8) | (int) buf[i];
    }

    fseek(fp, 14, SEEK_SET);

    return headersize;
}

/*
 * Read bitmap core header (OS/2 bitmap)
 */
BITMAPCOREHEADER *ReadBMCoreHeader(FILE *fp)
{
    BITMAPCOREHEADER *header;
    unsigned int   headersize;
    int            width;
    int            height;
    unsigned short planes;
    unsigned short bitcount;

    /* Header size (4 bytes) */
    headersize = (unsigned int) ReadLE4(fp);

    /* Width (2 bytes) */
    width = (int) ReadLE2(fp);

    /* Height (2 bytes) */
    height = (int) ReadLE2(fp);
```

```
    /* Planes (2 bytes) */
    planes = (unsigned short) ReadLE2(fp);

    /* Bit Count (2 bytes) */
    bitcount = (unsigned short) ReadLE2(fp);

    header = (BITMAPCOREHEADER *) malloc(sizeof(BITMAPCOREHEADER));
    header->bcSize     = headersize;
    header->bcWidth    = width;
    header->bcHeight   = height;
    header->bcPlanes   = planes;
    header->bcBitCount = bitcount;

    return header;
}

/*
 * Read bitmap info header (Windows bitmap)
 */
BITMAPINFOHEADER *ReadBMInfoHeader(FILE *fp)
{
    BITMAPINFOHEADER *header;
    unsigned int   headersize;
    int            width;
    int            height;
    unsigned short planes;
    unsigned short bitcount;
    unsigned int   compression;
    unsigned int   size_image;
    int            x_pix_per_meter;
    int            y_pix_per_meter;
    unsigned int   clr_used;
    unsigned int   clr_important;

    /* Header size (4 bytes) */
    headersize = (unsigned int) ReadLE4(fp);

    /* Width (4 bytes) */
    width = (int) ReadLE4(fp);

    /* Height (4 bytes) */
    height = (int) ReadLE4(fp);
```

```
    /* Planes (2 bytes) */
    planes = (unsigned short) ReadLE2(fp);

    /* Bit Count (2 bytes) */
    bitcount = (unsigned short) ReadLE2(fp);

    /* Compression (4 bytes) */
    compression = (unsigned int) ReadLE4(fp);

    /* Size image (4 bytes) */
    size_image = (unsigned int) ReadLE4(fp);

    /* X pix per meter (4 bytes) */
    x_pix_per_meter = (int) ReadLE4(fp);

    /* Y pix per meter (4 bytes) */
    y_pix_per_meter = (int) ReadLE4(fp);

    /* Color used (4 bytes) */
    clr_used = (unsigned int) ReadLE4(fp);

    /* Color important (4 bytes) */
    clr_important = (unsigned int) ReadLE4(fp);

    header = (BITMAPINFOHEADER *) malloc(sizeof(BITMAPINFOHEADER));
    header->biSize         = headersize;
    header->biWidth        = width;
    header->biHeight       = height;
    header->biPlanes       = planes;
    header->biBitCount     = bitcount;
    header->biCompression  = compression;
    header->biSizeImage    = size_image;
    header->biXPixPerMeter = x_pix_per_meter;
    header->biYPixPerMeter = y_pix_per_meter;
    header->biClrUsed      = clr_used;
    header->biClrImportant = clr_important;

    return header;
}

/*
 * Read 2 bytes in little endian
 */
unsigned short ReadLE2(FILE *fp)
```

```
{
    unsigned char buf[2];
    unsigned short result = 0;
    int i;

    fread(buf, 1, 2, fp);
    for (i = 1; i >= 0; i--) {
        result = (result << 8) | (unsigned short) buf[i];
    }

    return result;
}

/*
 * Read 4 bytes in little endian
 */
unsigned int ReadLE4(FILE *fp)
{
    unsigned char buf[4];
    unsigned int result = 0;
    int i;

    fread(buf, 1, 4, fp);
    for (i = 3; i >= 0; i--) {
        result = (result << 8) | (unsigned int) buf[i];
    }

    return result;
}
```

2.Reading bitmap file informations.

```
typedef struct tagBITMAPFILEHEADER {
    char          bfType[3];    /* 2 bytes + null char */
    unsigned int   bfSize;       /* 4 bytes */
    unsigned short bfReserved1;  /* 2 bytes */
    unsigned short bfReserved2;  /* 2 bytes */
    unsigned long  bfOffBits;    /* 4 bytes */
} BITMAPFILEHEADER;

/*
 * Bitmap info header (Windows)
 */
typedef struct tagBITMAPINFOHEADER {
    unsigned int   biSize;          /* 4 bytes */
    long           biWidth;         /* 4 bytes */
    long           biHeight;        /* 4 bytes */
    unsigned short biPlanes;        /* 2 bytes */
    unsigned short biBitCount;      /* 2 bytes */
    unsigned int   biCompression;   /* 4 bytes */
    unsigned int   biSizeImage;     /* 4 bytes */
    long           biXPixPerMeter;  /* 4 bytes */
    long           biYPixPerMeter;  /* 4 bytes */
    unsigned long  biClrUsed;       /* 4 bytes */
    unsigned long  biClrImportant;  /* 4 bytes */
} BITMAPINFOHEADER;

/*
 * Bitmap core header (OS/2)
 */
typedef struct tagBITMAPCOREHEADER {
    unsigned int   bcSize;      /* 4 bytes */
    short          bcWidth;     /* 2 bytes */
    short          bcHeight;    /* 2 bytes */
    unsigned short bcPlanes;    /* 2 bytes */
    unsigned short bcBitCount;  /* 2 bytes */
} BITMAPCOREHEADER;


BITMAPFILEHEADER *ReadBMFileHeader(FILE *fp);
BITMAPINFOHEADER *ReadBMInfoHeader(FILE *fp);
BITMAPCOREHEADER *ReadBMCoreHeader(FILE *fp);

int SizeOfInformationHeader(FILE *fp);
```

3.Read and display BMP image informations.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "bmp.h"


int main(int argc, char *argv[])
{
    FILE *fp;
    BITMAPFILEHEADER *bmFileHeader = NULL;
    BITMAPCOREHEADER *bmCoreHeader = NULL;
    BITMAPINFOHEADER *bmInfoHeader = NULL;
    int headersize;

    if (argc != 2) {
        printf("Usage: bmpinfo <file.bmp>\n\n");
        exit(1);
    }

    if ((fp = fopen(argv[1], "rb")) == NULL) {
        printf("Cannot open file: %s\n\n", argv[1]);
        exit(1);
    }
    bmFileHeader = ReadBMFileHeader(fp);
    if (strcmp(bmFileHeader->bfType, "BM") != 0) {
        printf("The file is not BITMAP.\n");
        exit(1);
    }
    headersize = SizeOfInformationHeader(fp);
    if (headersize == 12) {
        bmCoreHeader = ReadBMCoreHeader(fp);
    } else if (headersize == 40) {
```

```
        bmInfoHeader = ReadBMInfoHeader(fp);
    } else {
        printf("Unsupported BITMAP.\n");
        exit(1);
    }
    fclose(fp);

    printf("File type          = %s\n", bmFileHeader->bfType);
    printf("File size          = %d bytes\n", bmFileHeader->bfSize);
    printf("Data offset        = %ld bytes\n", bmFileHeader->bfOffBits);
    if (headersize == 12) {
        printf("Info header size   = %d bytes\n", bmCoreHeader->bcSize);
        printf("Width              = %d pixels\n", bmCoreHeader->bcWidth);
        printf("Height             = %d pixels\n", bmCoreHeader->bcHeight);
        printf("Planes             = %d\n", bmCoreHeader->bcPlanes);
        printf("Bit count          = %d bits/pixel\n", bmCoreHeader->bcBitCount);
    } else if (headersize == 40) {
        printf("Info header size   = %d bytes\n", bmInfoHeader->biSize);
        printf("Width              = %ld pixels\n", bmInfoHeader->biWidth);
        printf("Height             = %ld pixels\n", bmInfoHeader->biHeight);
        printf("Planes             = %d\n", bmInfoHeader->biPlanes);
        printf("Bit count          = %d bits/pixel\n", bmInfoHeader->biBitCount);
        printf("Compression        = %d\n", bmInfoHeader->biCompression);
        printf("Size image         = %d bytes\n", bmInfoHeader->biSizeImage);
        printf("X pixels per meter = %ld\n", bmInfoHeader->biXPixPerMeter);
        printf("Y pixels per meter = %ld\n", bmInfoHeader->biYPixPerMeter);
        printf("Color used         = %ld colors\n", bmInfoHeader->biClrUsed);
    }

    free(bmFileHeader);
    free(bmCoreHeader);
    free(bmInfoHeader);

    return 0;
}
```