

Cソースコード1\_5\_改.txt

```
// _GRT_SECURE_NO_WARNINGS
```

```
//*****
```

```
/// ABHB_basic_form.exe
```

```
/// ABHB dll群実行基本フォーマット ソースコード  
/// dll群を実行するためのメモリ確保、ハンドルの受け渡しの為の基本実行ファイ  
ル サンプルソースコード
```

```
/// ABHB_basic_form.exe <char* result_dir> <char* file_name_source> <int  
width_pixel> <int height_pixel>
```

```
/// char* result_dir : 処理後の画像をダンプするdir  
/// char* file_name_source : クエリ画像 (bmp) ファイル名 フルパス  
/// int width_pixel : 画像横方向ピクセル数  
/// int height_pixel : 画像縦方向ピクセル数
```

```
/// version 1.0.0
```

```
/// ForgeVision Confidential
```

```
/// 2024.04.27
```

```
//*****
```

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <io.h>  
#include <time.h>  
#include <math.h>  
#include <windows.h>  
#include <direct.h>
```

```
#include "ABHB.h"
```

```
#pragma warning (disable: 6385; disable: 6386; disable: 6031; disable: 4996)
```

```
#define VERSION "version1.0.0"
```

```
// 関数定義
```

```
void usage(char* cmdname);
```

```
int main_process (unsigned char** gRed, unsigned char** gGreen, unsigned char** gBlue,  
unsigned char** gBuff_red, unsigned char** gBuff_green, unsigned char**  
gBuff_blue,  
int width_pixel, int height_pixel);
```

```
/**  
* メイン関数 *  
**
```

## Cソースコード1\_5\_改.txt

```

int main(int argc, char* argv[])
{
    // 引数情報を読み込みます
    int return_code, i;
    int width_pixel_org, height_pixel_org, width_pixel, height_pixel;
    char dump_dir[256], file_name_source[256], file_name_dump[256];
    clock_t gStart_time, gEnd_time;

    // 引数の項目数が不足している場合はUsageを表示してエラー終了します
    if (argc < 5) {
        usage(argv[0]);
        Sleep(10);
        return -1;
    }

    // 引数を変数に代入します
    strcpy(dump_dir, argv[1]);
    strcpy(file_name_source, argv[2]);
    width_pixel_org = atoi(argv[3]);
    height_pixel_org = atoi(argv[4]);

    printf("%s¥n", VERSION);

    // 現在時刻取得
    gStart_time = clock();

    // 2次元配列、gRed[][]、gGreen[][]、gBlue[][]、gBuff_red[][]、gBuff_green[][],
    gBuff_blue[][]のメモリ確保
    unsigned char** gRed = (unsigned char**)malloc(sizeof(unsigned char*) *
height_pixel_org);
    unsigned char* gRed_p = (unsigned char*)malloc(sizeof(unsigned char) *
height_pixel_org * width_pixel_org);
    for (i = 0; i < height_pixel_org; i++) {
        gRed[i] = gRed_p + (long long)i * (long long)width_pixel_org;
    }

    unsigned char** gGreen = (unsigned char**)malloc(sizeof(unsigned char*) *
height_pixel_org);
    unsigned char* gGreen_p = (unsigned char*)malloc(sizeof(unsigned char) *
height_pixel_org * width_pixel_org);
    for (i = 0; i < height_pixel_org; i++) {
        gGreen[i] = gGreen_p + (long long)i * (long long)width_pixel_org;
    }

    unsigned char** gBlue = (unsigned char**)malloc(sizeof(unsigned char*) *
height_pixel_org);
    unsigned char* gBlue_p = (unsigned char*)malloc(sizeof(unsigned char) *
height_pixel_org * width_pixel_org);
    for (i = 0; i < height_pixel_org; i++) {
        gBlue[i] = gBlue_p + (long long)i * (long long)width_pixel_org;
    }

    unsigned char** gBuff_red = (unsigned char**)malloc(sizeof(unsigned char*) *
height_pixel_org);
    unsigned char* gBuff_red_p = (unsigned char*)malloc(sizeof(unsigned char) *
height_pixel_org * width_pixel_org);
    for (i = 0; i < height_pixel_org; i++) {
        gBuff_red[i] = gBuff_red_p + (long long)i * (long long)width_pixel_org;
    }

    unsigned char** gBuff_green = (unsigned char**)malloc(sizeof(unsigned char*) *
height_pixel_org);

```

Cソースコード1\_5\_改.txt

```

    unsigned char* gBuff_green_p = (unsigned char*)malloc(sizeof(unsigned char) *
height_pixel_org * width_pixel_org);
    for (i = 0; i < height_pixel_org; i++) {
        gBuff_green[i] = gBuff_green_p + (long long)i * (long
long)width_pixel_org;
    }

    unsigned char** gBuff_blue = (unsigned char**)malloc(sizeof(unsigned char*) *
height_pixel_org);
    unsigned char* gBuff_blue_p = (unsigned char*)malloc(sizeof(unsigned char) *
height_pixel_org * width_pixel_org);
    for (i = 0; i < height_pixel_org; i++) {
        gBuff_blue[i] = gBuff_blue_p + (long long)i * (long
long)width_pixel_org;
    }

    // bmpファイルを読み込み、RGB情報をそれぞれgRed[][], gGreen[][], gBlue[][]に格
納します
    width_pixel = width_pixel_org;
    height_pixel = height_pixel_org;
    return_code = ABHB_bmp_read_write(0, file_name_source, gRed, gGreen, gBlue,
&width_pixel, &height_pixel);

    // 処理を実行します
    return_code = main_process(gRed, gGreen, gBlue, gBuff_red, gBuff_green,
gBuff_blue, width_pixel, height_pixel);

    // gBuff_red[][], gBuff_green[][], gBuff_blue[][]の情報をRGBの値とした結果bmpフ
ァイルを書き出します
    sprintf(file_name_dump, "%s/result.bmp", dump_dir);
    return_code = ABHB_bmp_read_write(1, file_name_dump, gBuff_red, gBuff_green,
gBuff_blue, &width_pixel, &height_pixel);

    // メモリ開放
    free(gBuff_blue_p);
    free(gBuff_blue);
    free(gBuff_green_p);
    free(gBuff_green);
    free(gBuff_red_p);
    free(gBuff_red);
    free(gBlue_p);
    free(gBlue);
    free(gGreen_p);
    free(gGreen);
    free(gRed_p);
    free(gRed);

    gEnd_time = clock();
    printf("%n正常終了   処理時間 : %dmSec%n%n", gEnd_time - gStart_time);

    return 0;
}

void usage(char* cmdname)
{
    printf(
        "%n"
        "Usage : %s%n"
        "<.exe> <char* result_dir> <char* file_name_source> <int width_pixel>
<int height_pixel>%n"
        "   char* result_dir       : 処理後の画像をダンプするdir%n"
        "   char* file_name_source   : 元画像ファイル名フルパス%n"
    );
}

```

```

                                Cソースコード1_5_改.txt
    "    int width_pixel      : 横方向ピクセル数¥n"
    "    int height_pixel    : 縦方向ピクセル数¥n",
cmdname);
    return;
}

/*****
/* main_process */
*****/

// gRed, gGreen, gBlueの2次元配列にクエリ画像のRGB情報が格納されています
// 同じサイズのgBuff_red, gBuff_green, gBuff_blueに結果画像情報を格納してリターンします

int main_process(unsigned char** gRed, unsigned char** gGreen, unsigned char** gBlue,
                unsigned char** gBuff_red, unsigned char** gBuff_green, unsigned char**
                gBuff_blue,
                int width_pixel, int height_pixel) {

    // ここに画像処理プログラムをコーディングします
    // 今回のサンプルプログラムでは、クエリ画像の情報をそのままバッファ2次元配列に
    コピーして終了します
    int i, j;
    for (i = 0; i < height_pixel; i++) {
        for (j = 0; j < width_pixel; j++) {
            gBuff_red[i][j] = gRed[i][j];
            gBuff_green[i][j] = gGreen[i][j];
            gBuff_blue[i][j] = gBlue[i][j];
        }
    }

    return 0;
}

```