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

// 表面のザラザラ感を数値化します
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) {

  // クエリの全ピクセルをHSLに変換してgBuff_xxx[][]に格納します
  int i, j;
  int hue;
  unsigned char sat, brt;
  double x, y, z;
  for (i = 0; i < height_pixel; i++) {
    for (j = 0; j < width_pixel; j++) {
      ABHB_rgb_to_hsl(0, &gRed[i][j], &gGreen[i][j], &gBlue[i][j], &hue, &sat, &brt,
&x, &y, &z); //①
      gBuff_red[i][j] = hue * 256 / 360;    // 0～359を0～255に変換します
      gBuff_green[i][j] = sat;
      gBuff_blue[i][j] = brt;
    }
  }

  // カレントディレクトリにhsl画像を出力します
  char dump_file_name[256];
  sprintf(dump_file_name, "hsl.bmp");
  ABHB_bmp_read_write(1, dump_file_name, gBuff_red, gBuff_green, gBuff_blue,
&width_pixel, &height_pixel);

  // ごはんもしくは豆腐分部を白にした結果画像を生成します、
  // ダンプしたhsl.bmpからsatが0～20且つbrtが60～をごはんもしくは豆腐部分のHSL範囲とし
ます
  int hue_min = 0;
  int hue_max = 360;
  int sat_min = 0;
  int sat_max = 20;
  int brt_min = 60;
  int brt_max = 256;
  int brt_min2 = 200;    // brtが200以上のピクセルは無条件に対象ピクセルとします
  for (i = 0; i < height_pixel; i++) {
    for (j = 0; j < width_pixel; j++) {
      if (hue_min <= gBuff_red[i][j] && gBuff_red[i][j] <= hue_max &&
        sat_min <= gBuff_green[i][j] && gBuff_green[i][j] <= sat_max &&
        brt_min <= gBuff_blue[i][j] && gBuff_blue[i][j] <= brt_max) {
        gBuff_red[i][j] = 255;
        gBuff_green[i][j] = 255;
        gBuff_blue[i][j] = 255;
      }
      else if (brt_min2 <= gBuff_blue[i][j]) {
        gBuff_red[i][j] = 255;
        gBuff_green[i][j] = 255;
        gBuff_blue[i][j] = 255;
      }
      else {
        gBuff_red[i][j] = 0;
        gBuff_green[i][j] = 0;
        gBuff_blue[i][j] = 0;
      }
    }
```

```
  }

  // カレントディレクトリにtarget画像を出力します
  sprintf(dump_file_name, "target.bmp");
  ABHB_bmp_read_write(1, dump_file_name, gBuff_red, gBuff_green, gBuff_blue,
&width_pixel, &height_pixel);



  // roughness構造体をセットします
  roughness_t roughness;
  roughness.low_percent = 5;    // 暗い方の結果値のヒストグラムの暗い方からの%
  roughness.mid_percent = 50;    // 中間の結果値のヒストグラムの暗い方からの%
  roughness.high_percent = 95;    // 明るい方の結果値のヒストグラムの暗い方からの%
  roughness.step = 1;      // ヒストグラムの明るさを算出する際の縦横ピクセル数
  roughness.low_brt = 0;      // 暗い方の結果値
  roughness.mid_brt = 0;      // 中間の結果値
  roughness.high_brt = 0;      // 明るい方の結果値


  // 豆腐部分は左上にあるとして、ABHB_roughness_analysisを呼び出します
  printf("<< TOFU >>¥n");
  int hist_tofu[256] = { 0 }; //②
  int return_code = ABHB_roughness_analysis(gRed, gGreen, gBlue,
    0, height_pixel / 2, 0, width_pixel / 2, gBuff_blue, 255, hist_tofu, &roughness);
//③

  // ヒストグラムと結果を表示します
  int k;
  for (k = 0; k < 256; k++) {
    int kk;
    if (k % 4 == 0 && 0 < hist_tofu[k]) {
      printf(" %3d  ", k);
      for (kk = 1; kk <= hist_tofu[k]; kk++) {
        printf("*");
      }
      printf("¥n");
    }
  }

  // roughness.high_brt - roughness.low_brt＝ヒストグラムの分散度が大きいほどザラザラしています④
  int result_roughness = roughness.high_brt - roughness.low_brt;

  printf("  hist_low=%d  hist_mid=%d  hist_high=%d  result_roughness=%d¥n",
    roughness.low_brt, roughness.mid_brt, roughness.high_brt, result_roughness);

  // ごはん部分は右半分にあるとして、ABHB_roughness_analysisを呼び出します⑤
  printf("<< RICE >>¥n");
  int hist_rice[256] = { 0 };
  return_code = ABHB_roughness_analysis(gRed, gGreen, gBlue,
    0, height_pixel - 1, width_pixel / 2, width_pixel - 1, gBuff_blue, 255, hist_rice,
&roughness);

  // ヒストグラムと結果を表示します
  for (k = 0; k < 256; k++) {
    int kk;
    if (k % 4 == 0 && 0 < hist_rice[k]) {
      printf(" %3d  ", k);
      for (kk = 1; kk <= hist_rice[k]; kk++) {
        printf("*");
      }
      printf("¥n");
```

```
    }
  }

  // roughness.high_brt - roughness.low_brt＝ヒストグラムの分散度が大きいほどザラザラしています
  result_roughness = roughness.high_brt - roughness.low_brt;

  printf("  hist_low=%d  hist_mid=%d  hist_high=%d  result_roughness=%d¥n",
    roughness.low_brt, roughness.mid_brt, roughness.high_brt, result_roughness);

  return 0;
}
```

****関数***
ABHB_roughness_analysis()
引数1：int area_info_counter_max…エリア情報格納数最大値（メモリ確保した数）
引数2：int* area_info_counter…エリア情報数ポインタ
引数3：int** area_info…エリア情報2次元配列ポインタ
引数4：int width_pixel…横方向ピクセル数
引数5：int height_pixel…縦方向ピクセル数
引数6：unsigned char** gBuff_red…トレースしたピクセルを255とします
引数7：unsigned char** gBuff_green…有効エリアを囲む矩形の辺を255とします
引数8：unsigned char** gBuff_blue…=255のピクセルの塊をトレースします
引数9：int(*condition_func)(int, int, int, int, int, int)…条件設定関数