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

// エッジ部分を検出します
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) {

  // gGray[][]メモリ確保①
  int i;
  unsigned char** gGray = (unsigned char**)malloc(sizeof(unsigned char*) *
height_pixel);
  if (gGray == NULL)  return -99;
  unsigned char* gGray_p = (unsigned char*)malloc(sizeof(unsigned char) * height_pixel
* width_pixel);
  if (gGray_p == NULL) {
    free(gGray);
    return -99;
  }
  for (i = 0; i < height_pixel; i++) {
    gGray[i] = gGray_p + (long long)i * (long long)width_pixel;
  }

  // グレースケールを生成します②
  int j;
  for (i = 0; i < height_pixel; i++) {
    for (j = 0; j < width_pixel; j++) {
      gGray[i][j] = (gRed[i][j] + gGreen[i][j] + gBlue[i][j]) / 3;
    }
  }

  // エッジ部分をブルーにします
  int value = 255;
  int diff_distance = 2;
  int brt_diff_th = 15;
  int return_code = ABHB_edge_mapping(gGray, gBuff_blue, width_pixel, height_pixel,
    value, diff_distance, brt_diff_th); //③



  free(gGray_p);
  free(gGray);

  return 0;
}



****関数***
ABHB_edge_mapping()
引数１：unsigned char** gGray   クエリ画像のグレースケール値が格納されている配列のポイ
ンタ
引数２：unsigned char** gBuff_blue      結果をマッピングする配列のポインタ
引数３：int width_pixel         横方向ピクセル数
引数４：int height_pixel                縦方向ピクセル数
引数５：int value                       結果をマッピングする値
引数６：int diff_distance               微分距離（ピクセル）
引数７：int brt_diff_th         2階微分結果閾値（この値以上をエッジとする）
```