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

// 色相補正を実行します
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②
  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);

  // 髪部分を白にした画像を生成します、③
  // ダンプしたhsl.bmpからhueが5～20且つsatが10～60且つbrtが20～85を髪分部のHSL範囲とし
ます
  int hue_min = 5;
  int hue_max = 20;
  int sat_min = 10;
  int sat_max = 60;
  int brt_min = 20;
  int brt_max = 85;
  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 {
        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);
```

```
  // 白にしたピクセルはHSLを変更してRGBを算出、その他のピクセルはクエリ画像のRGBをコピーします
  int target_hue = 350;      // hueは350°（紫寄りの赤）に変更
  double saturation_ratio = 0.8;  // satは80%として色をより淡くに変更
  double brightness_ratio = 1;
  for (i = 0; i < height_pixel; i++) {
    for (j = 0; j < width_pixel; j++) {

      if (gBuff_red[i][j] == 255) {
        ABHB_rgb_to_hsl(0, &gRed[i][j], &gGreen[i][j], &gBlue[i][j], &hue, &sat, &brt, &x, &y, &z);
        hue = target_hue;
        sat = int(sat * saturation_ratio);
        if (255 < sat)  sat = 255;
        brt = int(brt * brightness_ratio);
        if (255 < brt)  brt = 255;
        ABHB_rgb_to_hsl(1, &gBuff_red[i][j], &gBuff_green[i][j], &gBuff_blue[i][j], &hue, &sat, &brt, &x, &y, &z); //④
      }
      else {
        gBuff_red[i][j] = gRed[i][j];
        gBuff_green[i][j] = gGreen[i][j];
        gBuff_blue[i][j] = gBlue[i][j];
      }
    }
  }



  return 0;
}
```

****関数***

```
ABHB_rgb_to_hsl(1, &gBuff_red[i][j], &gBuff_green[i][j], &gBuff_blue[i][j], &hue, &sat, &brt, &x, &y, &z)
```

引数１：int mode　　mode=0：RGB→HLS　mode=1：HLS→RGB
引数２：unsigned char* red　　Redの値が格納されている変数のポインタ
引数３：unsigned char* green　　Greenの値が格納されている変数のポインタ
引数３：unsigned char* blue　　Blueの値が格納されている変数のポインタ
引数5：int* hue　　Hueの値を格納する変数のポインタ
引数6：unsigned char* sat　　Saturationの値を格納する変数のポインタ
引数7：unsigned char* brt　　Lightnessの値を格納する変数のポインタ
引数8：double* x　　三次元空間のＸ座標（-255～255）ポインタ
引数9：double* y　　三次元空間のＹ座標（-255～255）ポインタ
引数10：double* z　　三次元空間のＺ座標（0～255）ポインタ