プログラミング 3 第 8-11 回レポート課題

和田 小百合
学生番号: 36714143

2025 年 12 月 17 日

目次

1　課題 8-11

1.1　レポート内容

今までのプログラミング 3 の授業で習った c 言語の技術を用いて twitter の画像投稿のサイズシュミレーションを作りました。今回このような作品を作ろうと思ったきっかけは、twitter の画像投稿で特定のサイズ以上の比率の画像を投稿するとき、twitter の決められた比率に変更されてしまい、見せたい情報がトリミングされて表示されなくなってしまうのを何回も目撃したので、先にシュミレーションできたらなと思い今回の課題で製作しました。

1.2　どのような技術を使ったか

今回はファイル操作、画像操作をメインで行っているので、外部ライブラリを使用しました。このため、コンパイル方法は、gcc task8.c -o task8 -lm でコンパイルを行っています。

1.3　構造体について

構造体に写真の画像サイズの情報を持つ構造体、ASCII アートの情報をもつ構造体、トリミングする領域の構造体を製作しました。以下が今回のプログラムでしようしている構造体です。

```c
  //構造体
typedef struct {
    int width;
    int height;
    int channels;
    unsigned char *data;
    bool from_stb;
} Image;

typedef struct {
    char **lines;       // 各行へのポインタ
    int line_count;     // 行数
    int max_width;      // 最長行幅
} AsciiArt;

typedef struct {
    int crop_x;
    int crop_y;
    int crop_w;
    int crop_h;
} CropArea;
```

1.4　それぞれの関数について

今回自作した関数は以下のようになります。

```c
bool load_image(const char *filename, Image *img);
AsciiArt load_ascii(const char *path);

CropArea compute_twitter_crop(Image *img);
```

```
5
6  Image ascii_to_image(const AsciiArt *ascii, int char_w, int char_h);
7  Image crop_image(const Image *src, const CropArea *crop);
8  Image concat_vertical(const Image *top, const Image *bottom);
9
10 void free_image(Image *img);
11 void free_ascii(AsciiArt *art);
12 bool save_png(const char *filename, const Image *img);
```

1.4.1 load_image について

load_image 関数は引数として、読み込むファイルの名前と、構造体 Image のポインタを取り、外部ライブラリを用いて画像から画像の幅・高さ・色の RGB を対応させる channel を main 関数にあるい img の情報を入れる関数です。

このような構造になっています。

```
1     bool load_image(const char *filename, Image *img) {
2       img->data = stbi_load(
3           filename,
4           &img->width,
5           &img->height,
6           &img->channels,
7           0
8       );
9
10      if (!img->data) {
11          fprintf(stderr, "画像の読み込みに失敗しました\n");
12          return false;
13      }
14      return true;
15  }
```

また、ファイルがなかった場合画像の読み込みに失敗しました。と表示するようにしています。

1.4.2 load_ascii 関数について

load_ascii 関数は、ascii アートのファイル名を取り込んで、AsciiArt の構造体にのポインタとの情報、行数、最長行幅の情報を取り入れます。この時、malloc 関数でファイルの操作で行数分のメモリを解放するようにしています。以下が関数の内容です。

```
1   // アート読み込みASCII
2 AsciiArt load_ascii(const char *path) {
3     AsciiArt art = {0};
4     // TODO: 行数を数える→で確保  malloc →読み込み
5     FILE *fp;
6     fp = fopen(path,"r");
7     int line_count  = 0;
8     char buffer[512];
9     //行数を数える
10    if(fp == NULL){      //ファイル操作
11        printf("ファイルを開けれませんでした。\n");
12        return art;
13    }
14    while(fgets(buffer,sizeof(buffer),fp)){
```

```
        line_count++;     //行数分カウント
    }
    fclose(fp);
    fp = fopen(path, "r");
    art.lines = malloc(sizeof(char*) * line_count);  //行のポインタ文のメモリを確保

    for(int i = 0; i < line_count ;i++){
        fgets(buffer,sizeof(buffer),fp);     //行の内容読み込み
        buffer[strcspn(buffer,"\n")] = '\0';
        int len = strlen(buffer);
        if(len > art.max_width){
            art.max_width = len;
        }
        art.lines[i] = malloc(len + 1);  //終端文字の分も確保\0
        strcpy(art.lines[i], buffer);
    }
    // ファイルを閉じる
    fclose(fp);
    art.line_count = line_count;

    return art;
}
```

1.4.3 compute_twitter_crop 関数について

compute_twitter_crop 関数は、構造体 img をもとに、画像が縦長、横長の場合に長さの短い方に合わせて、4:3,3:4 にトリミングする長さの情報を構造体の CropArea に情報に取り込み出力する関数です。

```
  CropArea compute_twitter_crop(Image *img ) {
    CropArea crop = {0};
    // TODO: 中心クロップの計算式を書く
    int height = img->height;
    int width = img->width;
    float aspect = (int)width/ height;


    //width, height は画像のピクセル数

    if(0.75 <= aspect && aspect <= 1.333){
        crop.crop_x = 0;
        crop.crop_y = 0;
        crop.crop_w = width;
        crop.crop_h = height;
        return crop;
    }
    if(aspect < 0.75f){   //縦長ちゃん
        //にトリミング4:3
        float target_ratio = 3.0f / 4.0f;  // 0.75
        crop.crop_w = width;     // 高さに合わせた幅
        crop.crop_h = (int)(width / target_ratio);
        crop.crop_x = 0;
        crop.crop_y = (height - crop.crop_h) / 2;
```

```
25
26      }else{
27          //にトリミング3:4,9:16
28          float target_ratio = 9.0f / 16.0f; // 0.5625
29          crop.crop_w = (int)(height * target_ratio);
30          crop.crop_h = height; // 幅に合わせた高さ
31          crop.crop_x = (width - crop.crop_w) / 2;
32          crop.crop_y = 0;
33      }
34
35      if (crop.crop_x < 0) crop.crop_x = 0;
36      if (crop.crop_y < 0) crop.crop_y = 0;
37      if (crop.crop_x + crop.crop_w > width)
38          crop.crop_w = width - crop.crop_x;
39      if (crop.crop_y + crop.crop_h > height)
40          crop.crop_h = height - crop.crop_y;
41
42      return crop;
43  }
```

4:3,9:16 の範囲内の画像はトリミングすることなく表示をする条件分岐を行い、その他を幅に合わせるか高さに合わせるかを変数をもちいて計算を行いました。

1.4.4 ascii_to_image 関数について

ascii_to_image 関数は、ascii アートを画像として読み込んでいく関数であり、ファイルからピクセルに変換して構造体 image にします。以下が関数の内容です。

```
1   Image ascii_to_image(const AsciiArt *ascii, int char_w, int char_h) {
2     Image img;
3     img.width  = ascii->max_width * char_w;
4     img.height = ascii->line_count * char_h;
5     img.channels = 3;
6     int offset_x = (img.width - ascii->max_width * char_w) / 2;
7
8
9     img.data = calloc(img.width * img.height * img.channels, 1);
10
11    for (int y = 0; y < ascii->line_count; y++) {
12        for (int x = 0; x < strlen(ascii->lines[y]); x++) {
13            if (ascii->lines[y][x] != ' ') {
14                for (int py = 0; py < char_h; py++) {
15                    for (int px = 0; px < char_w; px++) {
16                        int ix = (y * char_h + py) * img.width
17                               + (offset_x + x * char_w + px);
18                        img.data[ix * 3 + 0] = 255;
19                        img.data[ix * 3 + 1] = 255;
20                        img.data[ix * 3 + 2] = 255;
21                    }
22                }
23            }
24        }
25    }
26    return img;
```

```
}
```

また、のちに ascii アートの写真の img 構造体と、トリミングした写真の img の構造体を合わせる操作を行う関数があるので、トリミングする写真と色味の情報を合わせるため、rgb に統一するようにしています。

1.4.5 crop_image 関数について

crop_image 関数は、トリミングする場所の情報をもつ CropArea の構造体、実際の写真の画像サイズをもとに実際の写真をトリミングし png 保存するようになっています。以下が関数の内容です。

```

Image crop_image(const Image *src, const CropArea *crop) {
    Image out;
    out.width  = crop->crop_w;
    out.height = crop->crop_h;
    out.channels = src->channels;

    out.data = malloc(out.width * out.height * out.channels);
    if (!out.data) {
        out.width = out.height = 0;
        return out;
    }

    for (int y = 0; y < out.height; y++) {
        for (int x = 0; x < out.width; x++) {
            int src_x = crop->crop_x + x;
            int src_y = crop->crop_y + y;

            int src_i = (src_y * src->width + src_x) * src->channels;
            int dst_i = (y * out.width + x) * out.channels;

            for (int c = 0; c < src->channels; c++) {
                out.data[dst_i + c] = src->data[src_i + c];
            }
        }
    }
    return out;
}
```

1.4.6 concat_vertical 関数について

concat_vertical 関数は、ascii のアートの写真部分と、トリミングした写真を統合する関数です。以下が関数の内容です。

```

Image concat_vertical(const Image *top, const Image *bottom) {
    Image out;
    out.width = (top->width > bottom->width) ? top->width : bottom->width;
    out.height = top->height + bottom->height;
    out.channels = 3;

    out.data = calloc(out.width * out.height * out.channels, 1);

    // 上 () ASCII
```

```
    for (int y = 0; y < top->height; y++) {
        memcpy(
            &out.data[y * out.width * 3],
            &top->data[y * top->width * 3],
            top->width * 3
        );
    }

    // 下（画像）
    for (int y = 0; y < bottom->height; y++) {
        memcpy(
            &out.data[(y + top->height) * out.width * 3],
            &bottom->data[y * bottom->width * 3],
            bottom->width * 3
        );
    }


    return out;
}
```

その他の関数は png に保存する関数、メモリのデータ解放などを行っています。

1.5 実行例

実行手順について説明します。ascii アートを写真の上に表示する場合、yes, トリミングのみの表示は no で答え、写真のファイル ascii アートのパスを入力していき、トリミングされた写真を final.png を出力しました。出力結果はこのような感じになりました。

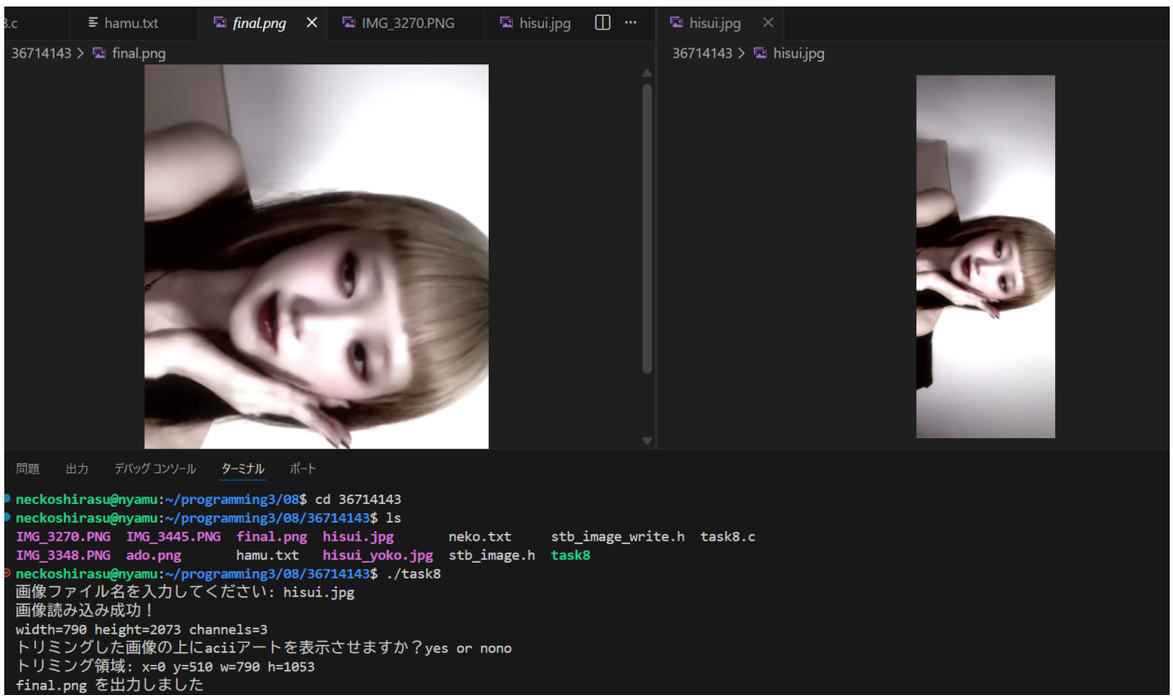

図 1　no/hisui.png

このように横長/縦長の画像についてちゃんと 4:3 または 3:4 にトリミングできることができました。

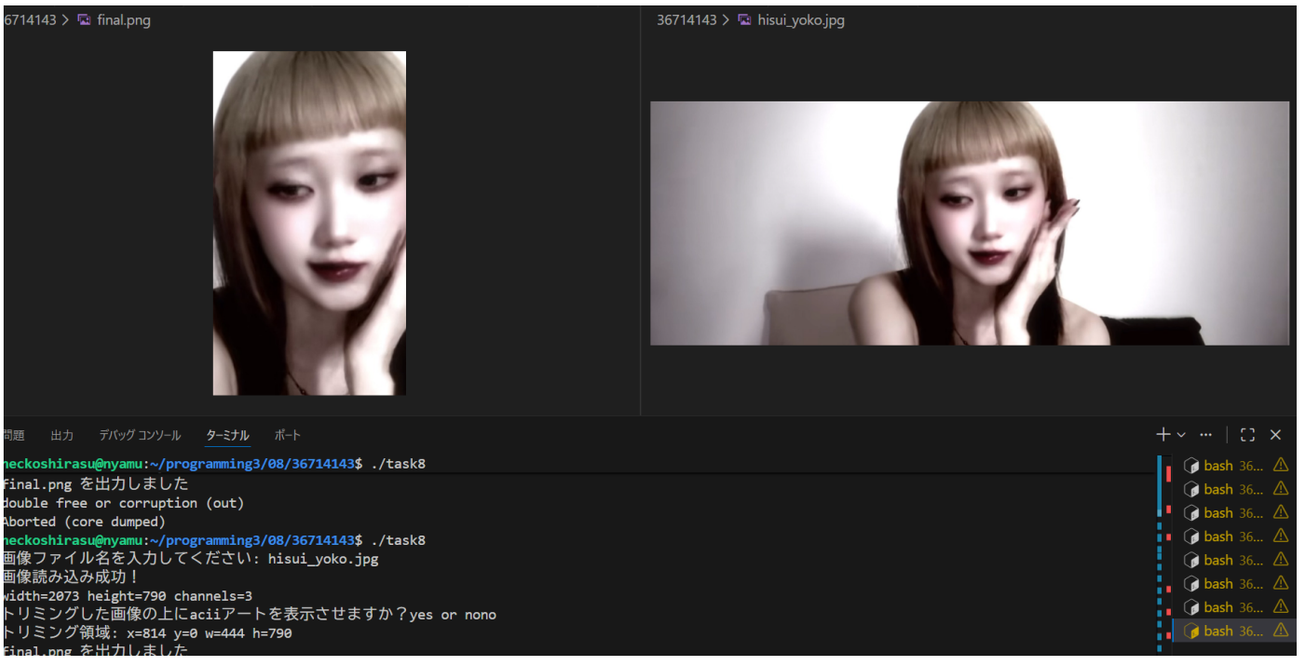

図 2　no/hisui_yoko.png

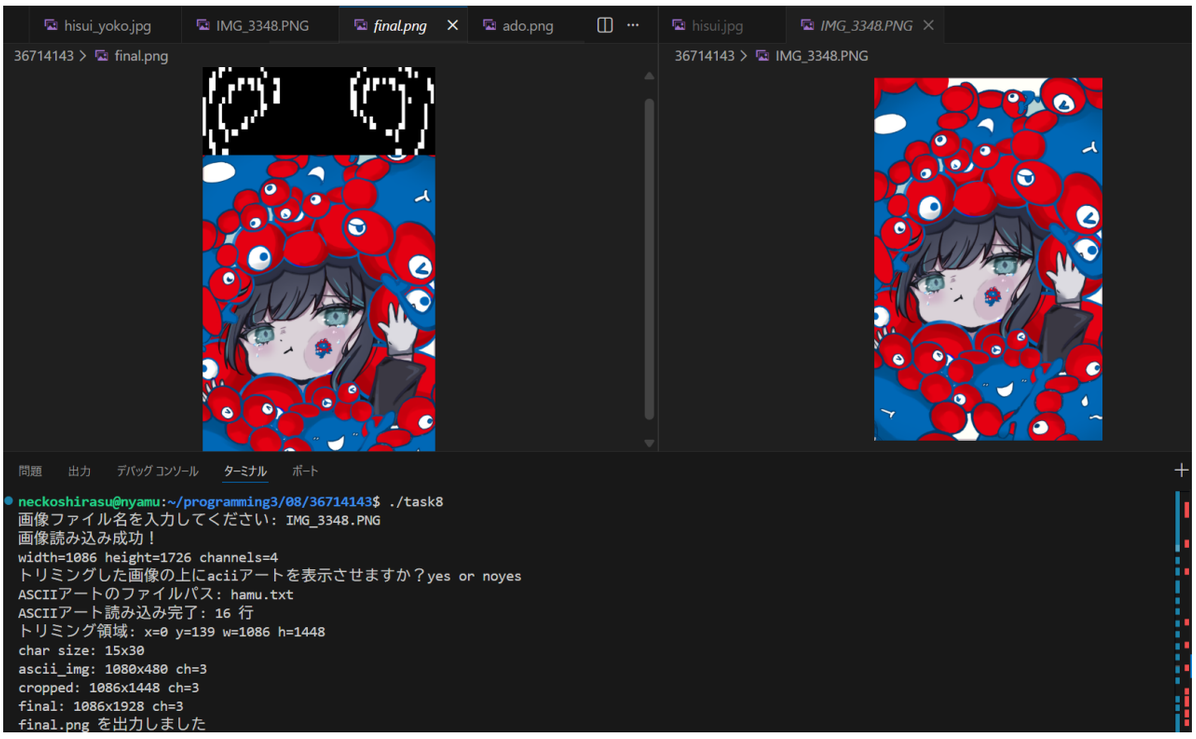

図 3　no/hisui_yoko.png

1.6　感想/課題点について

今回 twitter の投稿でアスキーアートを用いてかわいい投稿をするのがはやっていたのでできることならそれを再現しようと思いました。しかし、アスキーアートをそのまま流用してしまうと、画像サイズとの関係でほぼ表示されていないように見えてしまいました。また、twitter の投稿では縦長の場合は今回作成したコードと同じようなトリミングをしているのですが、横の仕様が様々であったため、また twitter のコードを見ながら改良していこうと思いました。