ファイル操作の基本

ファイル操作の基本

1. ファイルを開いてファイルポインタをセット
   fopen関数
2. ファイル操作用の関数を使用してデータを読み込む/書き込む
   fgets, fscanf, fread, fputs, fprintf, fwrite…
3. ファイルを閉じる
   fclose関数

ファイル読み込みの例

```
int main(void)
{
  FILE *fp;
  char lineBuffer[100];

  fp = fopen("input.txt", "r");

  fgets(lineBuffer, 100, fp);

  fclose(fp);

  return 0;
}
```

Hello

input.txt

ファイル読み込みの例

```
int main(void)
{
  FILE *fp;
  char lineBuffer[100];

  fp = fopen("input.txt", "r");

  fgets(lineBuffer, 100, fp);

  fclose(fp);

  return 0;
}
```

ファイルポインタの定義

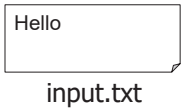
ファイル中のテキストを格納するための char配列

Hello

input.txt

ファイル読み込みの例

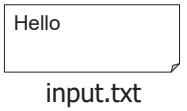
```
int main(void)
{
  FILE *fp;
  char lineBuffer[100];

  fp = fopen("input.txt", "r");

  fgets(lineBuffer, 100, fp);

  fclose(fp);

  return 0;
}
```



input.txtファイルを"read"モードで開き，ファイルポインタfpに関連付け

ファイル読み込みの例

```
int main(void)
{
  FILE *fp;
  char lineBuffer[100];

  fp = fopen("input.txt", "r");

  fgets(lineBuffer, 100, fp);

  fclose(fp);

  return 0;
}
```



【fopen関数】
第一引数：ファイル名
第二引数：モード
"r": テキストファイルの読み込み
"w": テキストファイルへの書き込み
"a": テキストファイルへの追加書き込み
"rb", "wb", "ab": バイナリファイル用の読み込み/書き込み/追加
（画像ファイルはバイナリファイル）

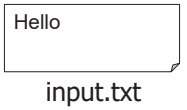
ファイル読み込みの例

```
int main(void)
{
  FILE *fp;
  char lineBuffer[100];

  fp = fopen("input.txt", "r");

  fgets(lineBuffer, 100, fp);

  fclose(fp);

  return 0;
}
```

fpに関連付けられたファイル（input.txt）から一行だけlineBufferに読み込む．
ただし読み込み上限は100バイトとする．
→lineBufferに'H' 'e' 'l' 'l' 'o' '¥0'が格納される

ファイル読み込みの例

```
int main(void)
{
  FILE *fp;
  char lineBuffer[100];

  fp = fopen("input.txt", "r");

  fgets(lineBuffer, 100, fp);

  fclose(fp);

  return 0;
}
```



【fgets関数】
第一引数：配列の先頭アドレス
第二引数：読み込み上限のバイト数
第三引数：ファイルポインタ

ファイル読み込みの例

```
int main(void)
{
  FILE *fp;
  char lineBuffer[100];

  fp = fopen("input.txt", "r");

  fgets(lineBuffer, 100, fp);

  fclose(fp);

  return 0;
}
```

ファイルクローズ

input.txt

ファイル書き込みの例

```
int main(void)
{
  FILE *fpo;
  char lineBuffer[100] = “Hello”;

  fpo = fopen("output.txt", "w");

  fprintf(fpo, "%s¥n", lineBuffer);

  fclose(fpo);

  return 0;
}
```

output.txt

ファイル書き込みの例

```
int main(void)
{
  FILE *fpo;
  char lineBuffer[100] = “Hello”;

  fpo = fopen("output.txt", "w");

  fprintf(fpo, "%s¥n", lineBuffer);

  fclose(fpo);

  return 0;
}
```

output.txtをwriteモードで開く

output.txt

ファイル書き込みの例

```
int main(void)
{
  FILE *fpo;
  char lineBuffer[100] = “Hello”;

  fpo = fopen("output.txt", "w");

  fprintf(fpo, "%s¥n", lineBuffer);

  fclose(fpo);

  return 0;
}
```

【fprintf関数】
第一引数：出力先のファイルポインタ．
他の引数はprintf関数と同じ書式．
つまり，output.txtに”Hello”が
書き込まれる

output.txt

ファイル書き込みの例

```
int main(void)
{
  FILE *fpo;
  char lineBuffer[100] = “Hello”;

  fpo = fopen("output.txt", "w");

  fprintf(fpo, "%s¥n", lineBuffer);

  fclose(fpo);

  return 0;
}
```

ファイルクローズ

Hello

output.txt

ファイル操作のまとめ

- ファイルポインタの宣言，open, closeは定型句
  - ほとんど考える必要がない
  - ファイル名とモード指定のみ注意
- ファイルへの文字列の入出力はコマンドラインの入出力関数（scanfやprintf）とほぼ同じ
  - ファイルポインタの指定を忘れずに

画像ファイルの操作

プログラムの流れ

```
int main(int argc, char* argv[])
{
  IMAGE in_image_data, out_image_data;

  /* 画像情報のロード */
  if (iioLoadFile(&in_image_data, "img01.ppm")) {
    return 1;
  }
  /* 画像情報のコピー */
  ipCopy(&in_image_data, &out_image_data);
  /* 画像情報の書き出し */
  iioSaveFile(&out_image_data, "header.txt");
  /* メモリリークチェック */
  _CrtDumpMemoryLeaks();
  return 0;
}
```

構造体の変数の宣言

画像ファイルからヘッダ情報を
in_image_dataに
読み込む関数

in_image_dataの画像情報を
out_image_dataにコピーする関数

out_image_dataをテキスト
ファイルに出力する関数

プログラムの流れ

- iioLoadFileの目的
  - ファイル名”fname”のPPMファイルを開き，画像情報をpImageに格納する
  - ひな形ではヘッダ情報の読み込み部分を作成済み
  - ヘッダ情報の読み込みは少々ややこしいので，以下はあくまで参考情報

```
int iioLoadFile(IMAGE* pImage, const char* fname)
{
  FILE* fpr;
  char lineBuffer[LINEMAX];
  int i;
```


【参考】プログラムの流れ

```
/* ファイルオープン */
if ((fpr = fopen(fname, "rb")) == NULL) {
  perror(fname);
  return 1;
}

/* ヘッダ部読み込み開始 */
fgets(lineBuffer, LINEMAX, fpr);

if (strcmp(lineBuffer, "P6¥n")) {
  // ERROR
  fclose(fpr);
  return 1;
}
```

ファイル名“fname”をreadモードで開く．ただし画像ファイルなので，バイナリモード”rb”にする

【参考】プログラムの流れ

```
/* ファイルオープン */
if ((fpr = fopen(fname, "rb")) == NULL) {
  perror(fname);
  return 1;
}

/* ヘッダ部読み込み開始 */
fgets(lineBuffer, LINEMAX, fpr);

if (strcmp(lineBuffer, "P6¥n")) {
  // ERROR
  fclose(fpr);
  return 1;
}
```

もしファイルが存在しない場合（ファイル名を間違えたなど）は，fopenはNULLを返す
→fprにNULLが代入される

【参考】プログラムの流れ

```
/* ファイルオープン */
if ((fpr = fopen(fname, "rb")) == NULL) {
  perror(fname);
  return 1;
}

/* ヘッダ部読み込み開始 */
fgets(lineBuffer, LINEMAX, fpr);

if (strcmp(lineBuffer, "P6¥n")) {
  // ERROR
  fclose(fpr);
  return 1;
}
```

もしファイルが存在しない場合（ファイル名を間違えたなど）は，fopenはNULLを返す
→fprにNULLが代入される

NULLかどうかを判定し，Yesであれば処理を中断（return）する．

NULLでなければ，ifの中は実行されない．

【参考】プログラムの流れ

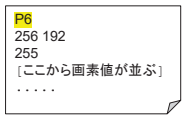
```
/* ファイルオープン */
if ((fpr = fopen(fname, "rb")) == NULL) {
  perror(fname);
  return 1;
}

/* ヘッダ部読み込み開始 */
fgets(lineBuffer, LINEMAX, fpr);

if (strcmp(lineBuffer, "P6¥n")) {
  // ERROR
  fclose(fpr);
  return 1;
}
```

perrorはエラー出力用の関数.
printfでも十分だが,
fopen失敗の直後に呼び出すと,
自動的に以下を出力するので便利

> fname: No such file or directory

【参考】プログラムの流れ

```
/* ファイルオープン */
if ((fpr = fopen(fname, "rb")) == NULL) {
  perror(fname);
  return 1;
}

/* ヘッダ部読み込み開始 */
fgets(lineBuffer, LINEMAX, fpr);

if (strcmp(lineBuffer, "P6¥n")) {
  // ERROR
  fclose(fpr);
  return 1;
}
```

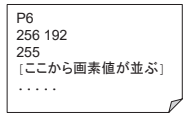


img01.ppm

PPMファイルから先頭1行を
読み込む.
つまり, lineBuffer ← "P6"

【参考】プログラムの流れ

```
/* ファイルオープン */
if ((fpr = fopen(fname, "rb")) == NULL) {
  perror(fname);
  return 1;
}

/* ヘッダ部読み込み開始 */
fgets(lineBuffer, LINEMAX, fpr);

if (strcmp(lineBuffer, "P6¥n")) {
  // ERROR
  fclose(fpr);
  return 1;
}
```



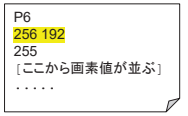
img01.ppm

lineBufferに読み込まれた文字列が
P6でなければPPMファイルではない
ので, if文の中を実行してreturn.

【参考】プログラムの流れ

```
fgets(lineBuffer, LINEMAX, fpr);

while (lineBuffer[0] == '#')
  fgets(lineBuffer, LINEMAX, fpr);

sscanf(lineBuffer, "%d %d¥n", &pImage->xsize, &pImage->ysize);

fgets(lineBuffer, LINEMAX, fpr);

sscanf(lineBuffer, "%d", &pImage->level);
```

次の1行をlineBufferに読み込む
lineBuffer ← "256 192"

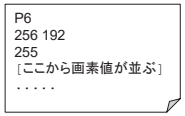


img01.ppm

【参考】プログラムの流れ

```
fgets(lineBuffer, LINEMAX, fpr);

while (lineBuffer[0] == '#')
  fgets(lineBuffer, LINEMAX, fpr);

sscanf(lineBuffer, "%d %d¥n", &plmage->xsize, &plmage->ysize);

fgets(lineBuffer, LINEMAX, fpr);

sscanf(lineBuffer, "%d", &plmage->level);
```

#から始まる文字列だった場合は
コメント文なので読み飛ばす
（今回は該当せず）

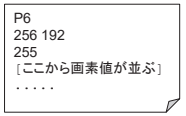


img01.ppm

【参考】プログラムの流れ

```
fgets(lineBuffer, LINEMAX, fpr);

while (lineBuffer[0] == '#')
  fgets(lineBuffer, LINEMAX, fpr);

sscanf(lineBuffer, "%d %d¥n", &plmage->xsize, &plmage->ysize);

fgets(lineBuffer, LINEMAX, fpr);

sscanf(lineBuffer, "%d", &plmage->level);
```

lineBufferの文字列"256 192"から
sscanfでxsizeとysizeを取得

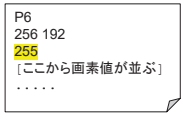


img01.ppm

【参考】プログラムの流れ

```
fgets(lineBuffer, LINEMAX, fpr);

while (lineBuffer[0] == '#')
  fgets(lineBuffer, LINEMAX, fpr);

sscanf(lineBuffer, "%d %d¥n", &plmage->xsize, &plmage->ysize);

fgets(lineBuffer, LINEMAX, fpr);

sscanf(lineBuffer, "%d", &plmage->level);
```

次の1行をlineBufferに読み込む
lineBuffer ← "255"

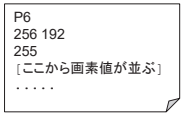


img01.ppm

【参考】プログラムの流れ

```
fgets(lineBuffer, LINEMAX, fpr);

while (lineBuffer[0] == '#')
  fgets(lineBuffer, LINEMAX, fpr);

sscanf(lineBuffer, "%d %d¥n", &plmage->xsize, &plmage->ysize);

fgets(lineBuffer, LINEMAX, fpr);

sscanf(lineBuffer, "%d", &plmage->level);
```

lineBufferの文字列"255"から
sscanfでlevelを取得



img01.ppm

【参考】プログラムの流れ

```
 /* バッファ用メモリ確保と画像
データの格納（2週目） */


 /* ファイルクローズ */
 fclose(fpr);

 return 0;
}
```

本来はここで画素値を読み込むが，
1週目は何もしない．
ヘッダ情報のみpImageに格納して終了．

ファイルクローズ

プログラムの流れ

```
int main(int argc, char* argv[])
{
 IMAGE in_image_data, out_image_data;

 /* 画像情報のロード */
 if (iioLoadFile(&in_image_data, "img01.ppm")) {
  return 1;
 }
 /* 画像情報のコピー */
 ipCopy(&in_image_data, &out_image_data);
 /* 画像情報の書き出し */
 iioSaveFile(&out_image_data, "header.txt");
 /* メモリリークチェック */
 _CrtDumpMemoryLeaks();
 return 0;
}
```

構造体の変数の宣言

画像ファイルからヘッダ情報を
in_image_dataに
読み込む関数

in_image_dataの画像情報を
out_image_dataにコピーする関数

【演習課題】
out_image_dataをテキスト
ファイルに出力する関数を
作成する

プログラムの流れ

- iioSaveFileの目的
  - （本来の目的）pImageの画像情報全てをファイル名”fname”のPPMファイルに出力する
  - （今週の目標）pImageのヘッダ情報のみをファイル名”fname”のTXTファイルに出力する

```
int iioSaveFile(IMAGE* pImage, const char* fname)
{
 int i;
 FILE* fpo;
```


【演習課題4_1_1】

- iioSaveFileのヘッダ出力部分を完成させる
  - “P6”と，pImageのxsize, ysize, levelを，PPMのヘッダ形式と同じようにテキストファイル”fname”に出力
  - header.txtが以下と同じ出力になるか確認
  - 最後は改行必須

```
P6
256 192
255
```

header.txt

```
int iioSaveFile(IMAGE* pImage, const char* fname)
{
 int i;
 FILE* fpo;

 /* ファイルオープン */

 /* PPMのヘッダ情報と画素値をファイルに出力 */

 /* ファイルクローズ */

 return 0;
}
```

演習課題に取り組む前に

- 演習課題ごとにファイルやプロジェクトのログを残しておくこと
  - できればプロジェクトのフォルダを丸ごとコピーして，演習課題番号をふっておく
  - そうしないと，間違って編集した際に1日目の最初からやり直すことになる

演習課題の目安: 20分