

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

1

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

2

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

3

4

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

} 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

5

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

} 【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)

6

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

} 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

7

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

} 【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

8

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

} ファイルクローズ

9

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

10

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

} output.txtをwriteモードで開く

11

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;
}
```

Hello
output.txt

} 【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

12

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;
}
```

Hello

output.txt

} ファイルクローズ

13

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

14

画像ファイルの操作

15

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

16

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plImage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* plImage, const char* fname)
{
    FILE* fpr;
    char lineBuffer[LINEMAX];
    int i;
```

17

【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

18

【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

19

【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。

20

【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;
}
```

P6
256 192
255
[ここから画素値が並ぶ]
.....

img01.ppm

} PPMファイルから先頭1行を
読み込む。
つまり、lineBuffer ← "P6"

21

22

【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;
}
```

P6
256 192
255
[ここから画素値が並ぶ]
.....

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", &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 ← "256 192"

P6
256 192
255
[ここから画素値が並ぶ]
.....

img01.ppm

23

24

【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);
```

#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)

```
P6  
256 192  
255  
[ここから画素値が並ぶ]  
.....
```

img01.ppm

25

【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);
```

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

```
P6  
256 192  
255  
[ここから画素値が並ぶ]  
.....
```

img01.ppm

26

【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 ← "255"

```
P6  
256 192  
255  
[ここから画素値が並ぶ]  
.....
```

img01.ppm

27

【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);
```

lineBufferの文字列"255"から
sscanfでlevelを取得

```
P6  
256 192  
255  
[ここから画素値が並ぶ]  
.....
```

img01.ppm

28

【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） */
```

} 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 */  
fclose(fpr);
```

} ファイルクローズ

```
return 0;  
}
```

29

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

30

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;
```

31

【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が下記の通り作成されるか確認

```
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;  
}
```

32

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