情報システムプログラミングⅡ（19回目）

2024年10月23日（水）

3～4限

授業内容

- 講義内容（教科書の530～547ページ）
  - ファイルの種類
  - ファイルの読み書きの基本
  - 1文字の読み書き
- 演習課題

ファイルの種類

■ファイル入出力

- 外部からのデータの読み込みおよび外部へのデータの書き込み（外部のファイルの読み書き）のこと

ファイルの種類

■コンピュータの扱うファイル

- 大きく分けると以下の2種類
  - テキストファイル：文字コードで解釈できるファイル（テキストエディタで正常に閲覧可能なファイル）
  - バイナリファイル：文字コードで解釈できないデータが含まれているファイル（テキストエディタで閲覧しようとすると文字化けするファイル）

ファイルの読み書きの基本

■ストリーム

- ファイル入出力の際，徐々に読み書きを行うことで，メモリ使用量の削減など効果的に入力と出力を行う仕組み

■標準入出力

- main関数の実行後から以下の3つのストリームを利用でき，入力はキーボード，出力はディスプレイに基本は接続される

名称	stdio.h でのマクロ定義	デフォルトの接続先
標準入力	stdin	キーボード
標準出力	stdout	ディスプレイ
標準エラー出力	stderr	ディスプレイ

ファイルの読み書きの基本

■ファイル入出力の流れ

- 大まかな流れは以下の4段階で，全て標準ライブラリに用意されている関数から処理できる
  1. FILE構造体へのポインタを宣言
  2. ファイルを開く
  3. 読み書きする
  4. ファイルを閉じる

■FILE構造体

- ファイル入出力に必要なメンバを持つ構造体
- FILE構造体へのポインタをファイルポインタという

ファイルの読み書きの基本

■fopen関数

- ファイルを開くための関数
- 戻り値はファイルポインタに格納して利用する

FILE* fopen(const char* filename, const char* mode)

filename：開きたいファイル名

mode　　：アクセスモード

戻り値　　：ファイルポインタ、エラー時は NULL

※ファイル名は、相対パスまたは絶対パスで指定する。

ファイルの読み書きの基本

■fopen関数

- アクセスモード（オープンモード）は以下の通り

アクセスモードでの「追加書き込み」は追記のこと

アクセスモードでの「書き込み」は上書きのこと

ファイルの種類	モード	意味	指定ファイルが存在するときの動作	指定ファイルが存在しないときの動作	読み書きの開始位置
テキスト	"r"	読み取り専用	ファイルを開く	エラー（戻り値 NULL）	先頭
	"r+"	読み書き			
	"w"	書き込み専用	サイズを 0 にしてファイルを開く	新規作成してファイルを開く	先頭
	"w+"	読み書き			
	"a"	追加書き込み専用	ファイルを開く	新規作成してファイルを開く	終端
	"a+"	読み取りと追加書き込み			
バイナリ	"rb"	読み取り専用	ファイルを開く	エラー（戻り値 NULL）	先頭
	"rb+" "r+b"	読み書き			
	"wb"	書き込み専用	サイズを 0 にしてファイルを開く	新規作成してファイルを開く	先頭
	"wb+" "w+b"	読み書き			
	"ab"	追加書き込み専用	ファイルを開く	新規作成してファイルを開く	終端
	"ab+" "a+b"	読み取りと追加書き込み			

ファイルの読み書きの基本

■fclose関数

- ファイルを閉じるための関数
- fopen関数を利用した場合は必ずfclose関数も利用すること

int fclose(FILE* fp)

fp　　：閉じるファイルのファイルポインタ

戻り値　：正常時は 0、エラー時は EOF (-1)

※ファイルポインタが NULL の場合の動作は保証されない。

ファイルの読み書きの基本

■ファイルを開いて閉じる例

ファイルポインタ名は
fpでなくてもよい

プログラムと同じ場所にある
「memo.txt」を読み込み専用で開く

ファイルを開けなかった
場合のエラー処理

```
#include <stdio.h>
#include <stdlib.h>  // exitを呼び出すために必要

int main(void)
{
  FILE* fp;                          (1)FILE 構造体へのポインタを宣言

  fp = fopen("memo.txt", "r");       (2)ファイルを開く
  if (fp == NULL) {
    printf("ファイルを開けませんでした¥n");
    exit(1);              // エラーなら異常終了
  } else {
    printf("ファイルを開きました¥n");
  }

  fclose(fp);                        (4)ファイルを閉じる

  return 0;
}
```

1文字の読み書き

■fputc関数

- 1文字（1バイトのデータ）を書き込む関数

int fputc(int ch, FILE* fp);

ch　　：書き込む文字
fp　　：書き込むファイルのファイルポインタ
戻り値：書き込んだ文字、失敗時は EOF (-1)

戻り値はint型なので，
文字での利用時は
char型にキャストする

■fgetc関数

- 1文字（1バイトのデータ）を読み込む関数

int fgetc(FILE* fp)

fp　　：読み取るファイルのファイルポインタ
戻り値：読み取った文字、ファイルの終端や失敗時は EOF (-1)

戻り値はint型なので，
文字での利用時は
char型にキャストする

1文字の読み書き

■1文字の読み書きの例

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
  FILE* fp;
  char text[] = "sukkriC!";      // 書き込む文字
  int len = strlen(text);        // 文字列の長さを取得
  int ch;

  // 書き込み専用でオープン
  if ((fp = fopen("memo.txt", "w")) == NULL) {
    exit(1);
  }

  for (int i = 0; i < len; i++) {
    fputc(text[i], fp);
  }

  fclose(fp);

  // 読み取り専用でオープン
  if ((fp = fopen("memo.txt", "r")) == NULL) {
    exit(1);
  }

  while ((ch = fgetc(fp)) != EOF) {
    putchar((char)ch);           // 標準出力（画面）に表示
  }

  fclose(fp);
  return 0;
}
```

エラー処理

コード 14-1 の 8 ～ 9 行目を 1 行で書くイディオム

配列の要素を 1 文字ずつ書き込む

エラー処理

ファイルを最後まで読んだらループ終了

char型にキャストしてから表示