# 第1回中間試験のお知らせ

- 5/30(金)に，第1回中間試験を実施します
  - 1講時（5511教室）：筆記試験
    - 筆記用具以外持ち込み不可
  - 2講時（5511教室）：実技試験
    - 教科書，参考書，ノート，講義資料は持ち込み可
    - 過去に作成したプログラムの閲覧やコピーは禁止，単位取得不可

- 範囲は，前半の講義内容全て
  - 再帰，ポインタ，文字列，構造体，ファイル操作
  - 演習問題，講義資料中の練習問題の復習をしてください

# データ構造入門及び演習
# 6回目:ファイル処理

2014/05/30

担当:見越　大樹
61号館304号室

# ファイル

- ファイルとは？
  - データやプログラムをディスク上に記録したもの


文書

画像


動画

- 大量のデータを扱うためには，ファイル処理が不可欠！

# ファイル処理とは?

- データ入力時:
  - キーボードからデータを入力
    → 入力間違い等があると効率が悪い
  - 記憶媒体上のファイルから入力する
    → 間違いの場合ファイルを修正できる

- データ出力時:
  - 計算結果等をディスプレイ上に表示する
    → 永久保存ができない
  - 記憶媒体上のファイルに出力する
    → 永久保存が可能

# ファイル処理の基本操作

- ファイルを扱うためには「ファイルポインタの宣言」が必要
  - FILEという名前の構造体を使用する
  - FILEというタグ(構造体名)はヘッダファイル“stdio.h”で定義済み
  - ファイル処理に必要な情報がメンバとして定義されている
- 宣言の方法：　FILE *fp;
- ファイルを扱う手順：
  1. ファイルを開く
  2. ファイルの読み書きを行う
  3. ファイルを閉じる

# ファイルを扱う手順 (1/3)

1. ファイルを開く
   - fopen()関数を使用する
   - 使用方法:

```
FILE  *fp;
fp = fopen( “test.txt”, “r” );
```

ファイルポインタ　ファイル名　モード

| モード | 意味 | ファイルがない時の動作 |
|---|---|---|
| “r” | 読み込み (read) | エラー |
| “w” | 書き出し (write) | ファイル作成 |
| “a” | 追記 (append) | ファイル作成 |
| “wb” | 書き出し (write) | ファイル作成 |
| “rb” | 読み込み (read) | エラー |

バイナリモードで読み書きする場合に使う（最近のコンパイラは使わなくても良い）

# ファイルを扱う手順 (2/3)

2. ファイルの読み書きを行う

- ファイルからのデータの読み込み：fscanf

  書式：fscanf ( ファイルポインタ，変換指定，実の引数 );

  fscanf( fp, “%d  %f”, &x, &y );

  ↑

  ファイルポインタfpからデータを読み込む

- ファイルへのデータの書き出し：fprintf

  書式：fprintf ( ファイルポインタ，変換指定，実の引数 );

  fprintf( fp, “%d  %f”, x, y );

  ↑

  ファイルポインタfpへデータが書き込まれる

# ファイルを扱う手順 (3/3)

3. ファイルを閉じる
   - モードにかかわらず，fclose()関数を使用する
   - 使用方法： fclose(fp);

# データの入出力例

ファイルからのデータの読み込み

```
#include <stdio.h>
int  main(void) {
    double  x, y, z;
    FILE  *fp;
    fp = fopen(“test1.txt”, “r”);
    fscanf( fp, “%f %f %f”, &x, &y, &z );
    fclose(fp);
    return  0;
}
```

ファイルへのデータの書き出し

```
#include <stdio.h>
int  main(void) {
    double  x=12.3, y=45.6, z=78.9;
    FILE  *fp;
    fp = fopen(“test2.txt”, “w”);
    fprintf( fp, “%f %f %f¥n”, x, y, z );
    fclose(fp);
    return  0;
}
```

# ファイルオープン時のエラー処理

- すでに存在するファイルをオープンする時に，ファイル名を間違えて指定した場合
  - ファイルが存在しないのでエラーが起こる
  - プログラムが異常終了しないように，エラー処理を行う
  - 読み込みモード(“r”)でファイルをオープンした場合，もしファイルがなければ，fopen()はファイルポインタに「NULL」を返す
  - ファイルが存在しないと判断した場合は，ファイルからデータを読み出さず，プログラムを強制終了する

# エラー処理の背景

- fopen()を実行後にファイルポインタを確認
  - ファイルポインタがNULLの場合，ファイルがオープンできない
  - NULL：ヘッダファイル<stdio.h>で既に定義されている

- プログラムの強制終了
  - exit()：プログラムを終了する関数
  - exit()：<stdlib.h>で定義されている
  - exit()の引数には1を入れる（exit(1)）

# エラー処理例

## エラー処理なし

```
#include <stdio.h>
int  main(void) {
    double  a, b, c;
    FILE  *fp;
    char  fname[64];
    printf( “Input file name >>” );
    scanf( “%s”, fname );
    fp = fopen( fname, “r” );
    fscanf( fp, “%f %f %f¥n”, &a, &b, &c );
    printf( “%f %f %f¥n”, a, b, c );
    fclose(fp);
    return  0;
}
```

## エラー処理あり

```
#include <stdio.h>
#include <stdlib.h>
int  main(void) {
    double  a, b, c;
    FILE  *fp;
    char  fname[64];
    printf( “Input file name >>” );
    scanf( “%s”, fname );
    fp = fopen( fname, “r” );
    if( fp == NULL ){
        printf(“Cannot open the file!!¥n”);
        exit(1);
    }
    fscanf( fp, “%f %f %f¥n”, &a, &b, &c );
    printf( “%f %f %f¥n”, a, b, c );
    fclose(fp);
    return  0;
}
```

# データの繰り返し書き出し

```
#include <stdio.h>
#include <math.h>
#define pai 3.1415

int main()
{
    int i;
    char fname[64];
    float rad, pitch;
    FILE *fp;

    printf("ファイルの名前を入力:");
    scanf("%s",fname);
                                          // ファイルのオープン

    pitch = 2.0*pai/360.0;                // ピッチの計算

    // データの繰り返し書き込み
    for(                          ){
        rad =
        fprintf(                              );
    }

    fclose(fp);                   // ファイルのクローズ

    return 0;
}
```

ファイルの中身

| | |
|---|---|
| 0 | 0.0000 |
| 1 | 0.0175 |
| 2 | 0.0349 |
| 3 | 0.0523 |
| 4 | 0.0698 |
| 5 | 0.0872 |
| 6 | 0.1045 |
| 7 | 0.1219 |
| 8 | 0.1392 |
| ⋮ | ⋮ |
| 354 | -0.1047 |
| 355 | -0.0873 |
| 356 | -0.0699 |
| 357 | -0.0525 |
| 358 | -0.0351 |
| 359 | -0.0176 |

# データの繰り返し読み込み

```
#include <stdio.h>
#include <stdlib.h>
#define NUM 360

int main()
{
    int i,x;
    char fname[64];
    float sin;
    FILE *fp;

    //  ファイル名の入力
    printf("ファイルの名前を入力:");
    scanf("%s",fname);

    fp =                                    //  ファイルのオープン

    if( fp == NULL ){
                printf("ファイルがオープンできません！¥n");
                exit(1);
    }

    //  データの繰り返し読み込み
    for (                          ){

                fscanf                              );
                printf(                             );
    }

    fclose(fp);                 //  ファイルのクローズ
    return 0;
}
```

ファイルの中身

| | |
|---|---|
| 0 | 0.0000 |
| 1 | 0.0175 |
| 2 | 0.0349 |
| 3 | 0.0523 |
| 4 | 0.0698 |
| 5 | 0.0872 |
| 6 | 0.1045 |
| 7 | 0.1219 |
| 8 | 0.1392 |
| ： | |
| 354 | -0.1047 |
| 355 | -0.0873 |
| 356 | -0.0699 |
| 357 | -0.0525 |
| 358 | -0.0351 |
| 359 | -0.0176 |

# その他の標準関数

## fgets()：ファイルからの1行読み込み

- ファイルの1行分を読み込み，末端の¥nをつける
- 使い方：

```
// ファイルを開く
FILE *fp;
fp = fopen( “file1.dat”, “r” );

// データを読み込む
char s[10];
fgets( s, 10, fp );
```

格納用文字配列　読み込み最大文字数　ファイルポインタ

```
// ファイルを閉じる
fclose(fp);
```

# その他の標準関数

fputs()：ファイルへの1行書き出し

- 文字配列の内容をファイルへ1行分書き出す
- 使い方：

```
//  ファイルを開く
FILE  *fp;
fp = fopen( “file1.dat”, “w” );

//  データを書き出す
fputs( “Hello!!¥n”, fp );

//  ファイルを閉じる
fclose(fp);
```

書き出す文字列

ファイルポインタ

# fgets()の使用例

```
#include <stdio.h>
#include <stdlib.h>
int  main(void)
{
      char  str[128], fname[64];
      FILE  *fp;
      int  i;
      printf( “Input file name >> ” );
      scanf( “%s”, fname );
      fp = fopen( fname, “r” );
      if ( fp == NULL ){
           printf( “Cannot open the file !” );
           exit(1);
      }
      for ( i=0;i<4;i++ ) {
           fgets( str, 128, fp ); // 1行読み込み
           printf( “%s”, str );
      }
      fclose(fp);
      return  0;
}
```

ファイルの中身(test2.txt)

```
abcdefg hijklmn opqr
stu vwxyz
I am a student.
You are not a student.
```

結果(ディスプレイ上の表示)

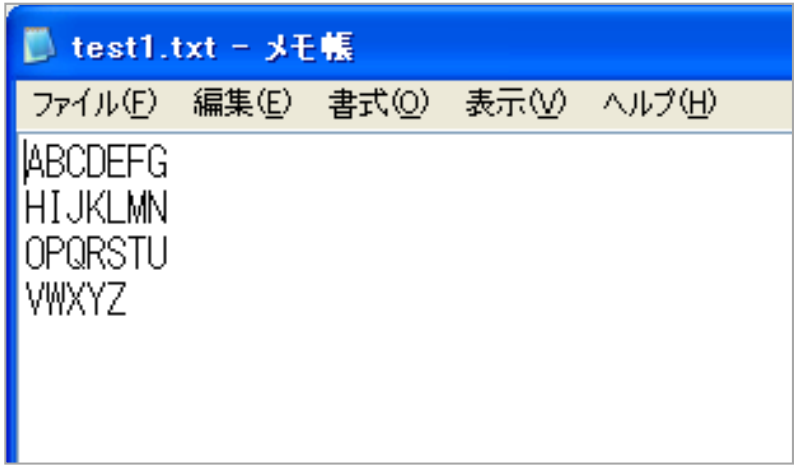
```
Input file name >> test2.txt
abcdefg hijklmn opqr
stu vwxyz
I am a student.
You are not a student.
```

```
Input file name >> test3.txt
Cannot open the file !
```

# fputs()の使用例

```
#include <stdio.h>
int main(void)
{
    char  *x[] = {
        { “ABCDEFG¥n” },
        { “HIJKLMN¥n” },
        { “OPQRSTU¥n” },
        { “VWXYZ¥n” }
    };
    FILE *fp;
    fp = fopen( “test1.txt”, “w” );
    int  i;
    for( i=0;i<4;i++ ){
        fputs( x[i], fp );     //  1行書き出し
    }
    fclose(fp);
    return 0;
}
```

結果
(test1.txtの中身)



# fread( ) : バイナリデータを読み込む

```
//  ファイルを開く
FILE  *fp;
fp = fopen( “file1.dat”, “r” );    //もしくは fp = fopen( “file1.dat”, “rb” );

//  データを読み込む
int buffer;
fread( &buffer, sizeof(buffer), 1, fp );
```

ファイルポインタ

格納用変数

読み込むサイズ

読み込む回数

```
char data[10];
fread( data, sizeof(char), 10, fp);
```

1byteを10回読み込む

```
//  ファイルを閉じる
fclose(fp);
```

# fwrite( ) : バイナリデータを書き込む

```
//  ファイルを開く
FILE  *fp;
fp = fopen( “file1.dat”, “r” );    //もしくは fp = fopen( “file1.dat”, “rb” );

//  データを読み込む
int buffer = 10;
fwrite( &buffer, sizeof(buffer), 1, fp );
```

ファイルポインタ

書き込むデータ

データサイズ

書き込む回数

```
char data[3] ={1, 2, 3};
fwrite( data, sizeof(char), 3, fp);
```

1byteを3回書き込む

```
//  ファイルを閉じる
fclose(fp);
```

# ファイルを扱う標準関数（まとめ）

- オープン/クローズ
  - fopen　　ファイルを開く
  - fclose　　ファイルを閉じる
- 入出力
  - fgets　　1行読み込む
  - fputs　　1行書き出す
  - fgetc　　1文字読み込む
  - fputc　　1文字書き出す
- 書式文字列付き入出力
  - fprintf　　書式文字列を指定してファイルに書き出す
  - fscanf　　書式文字列を指定してファイルから読み込む

# ファイル処理でよく使われる記号

- EOF ：ファイルの終わり
- NULL：ポインタがどのアドレスも指していない状態)
  - ファイルがないとき，fopen 関数から返される
  - ファイルが終わりに達すると，fgets関数から返される
- stdin：キーボード（標準入力）
- stdout：ディスプレイ（標準出力）
- stderr：標準エラー出力（通常標準出力に出力）

# ファイル処理手順のまとめ

0. 必要なヘッダファイルを追加する
   - #include <stdlib.h>　など
1. ファイルを開く
   - fp = fopen();
   - エラー処理を入れること
2. ファイルの読み書きを行う
3. ファイルを閉じる
   - fclose(fp);

# #include について

- #includeはただのファイル挿入＝＞ファイルの中身がそのまま挿入される
- 自分のディレクトリにある .h は，“　”で囲んでインクルードする

例）　file_process.c , file_process.h , program.cがある

```
#include “file_process.h”

int main(void){
    filereader(“filename.txt”);
     .
     .
     .
}
```

program.c

```
void filereader(char *name);
//  プロトタイプ宣言
```

file_process.h

```
void filereader(char *name){
    FILE * fp;
    int i;
    ……
} // 関数本体
```

file_process.c

# 分割コンパイル

- 規模の大きなソースコードを書く場合は、ソースファイルを分割してプログラミングする

例）

file_process.c, program.c　の2つに分けてある

main関数はprogram.cにだけ書いてある

gcc　file_process.c　とするとmain関数が無いためエラー

gcc　program.c　　もfile_process.c の関数を利用しているのでエラー

[正しい手順]

gcc –c file_process.c　// file_process.cをコンパイルして中間データを作成

gcc –c program.c　　// program.cをコンパイルして中間データを作成

gcc file_process.o　program.o　// 実行ファイルを生成