ファイル入出力

三池　克明

ファイルのデータを読込む、あるいは書込む方法と、CSV 形式のファイル処理について解説します。

—目　次—

1.　ファイルにアクセスするには

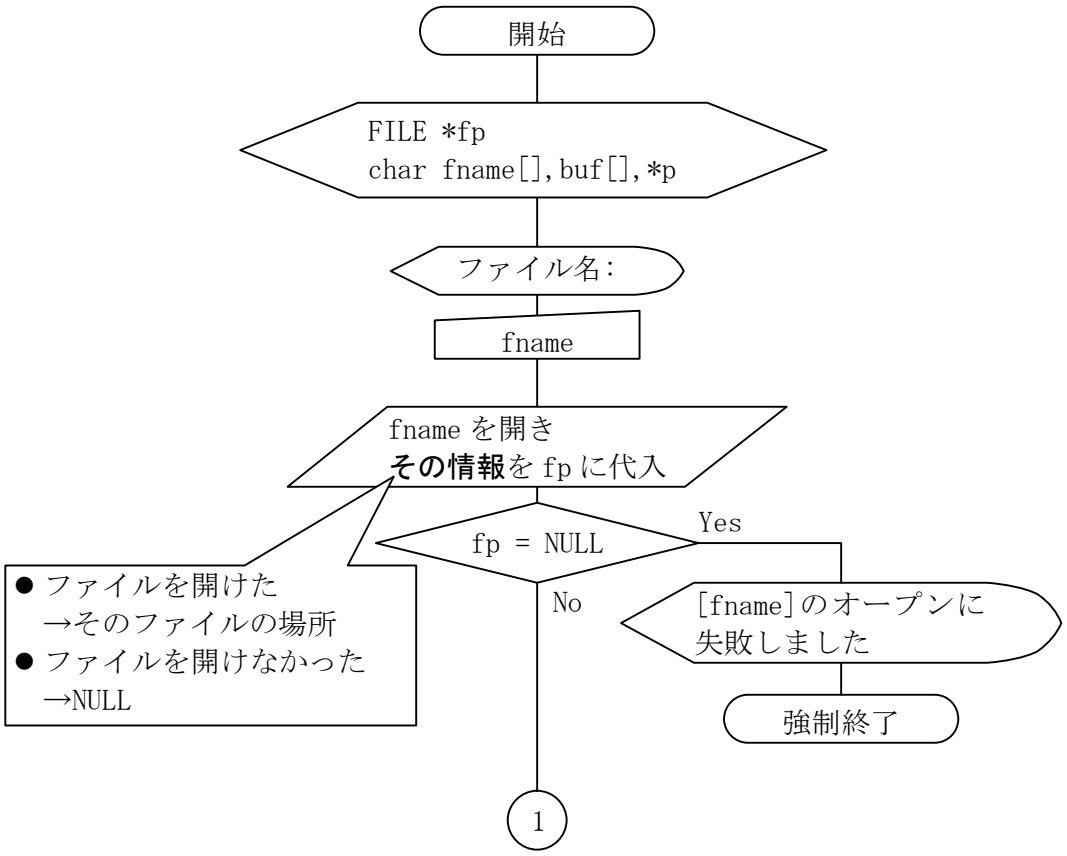
ファイルにアクセスするには、

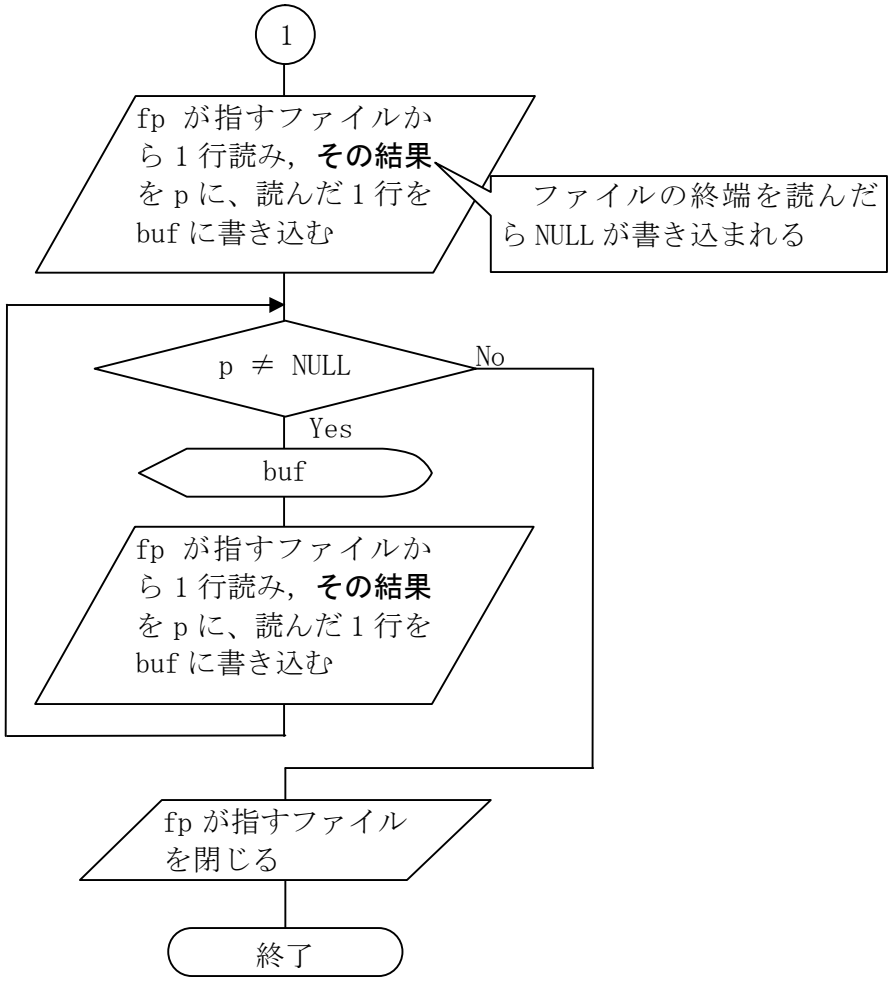
1. ファイルを開く
2. アクセスする
3. ファイルを閉じる

という手順を踏まなければなりません。

1.1. ファイルを読み込む

まずはファイルの内容を画面に表示させるプログラムを作りましょう。





fileread1.cpp

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define  NAMESIZE 256
 5:  #define LINESIZE 1024
 6:
 7:  main()
 8:  {
 9:          FILE    *fp;
10:          char    fname[NAMESIZE], buf[LINESIZE], *p;
11:
```

```
12:             printf("ファイル名 : ");
13:             gets(fname);
14:
15:             /*ファイルを開く*/
16:             fp = fopen(fname, "r");
17:             if(fp == NULL){
18:                     printf("%s のオープンに失敗しました。¥n", fname);
19:                     exit(1);
20:             }
21:
22:             /*一行ずつ読む*/
23:             p = fgets(buf, LINESIZE, fp);
24:             while(p != NULL){
25:                     printf("%s", buf);
26:                     p = fgets(buf, LINESIZE, fp);
27:             }
28:
29:             /*ファイルを閉じる*/
30:             fclose(fp);
31:     }
```

実行例１

```
ファイル名 : hello.cpp
/* Hello World! を表示する*/

#include <stdio.h>

main()
{
        printf("Hello World!¥n");
}
```

そのファイルが存在すればこのように画面に表示されます。
ただし、きちんと表示されるのはテキスト形式のファイルのみです。

実行例 2

```
ファイル名 : hoge
hoge のオープンに失敗しました。
```

一方、存在しないファイル名を入力するとエラーメッセージが表示されます。

それではソースプログラムをたどってみましょう。

```
 9:          FILE    *fp;
10:          char    fname[NAMESIZE], buf[LINESIZE], *p;
```

9 行目の「FILE *fp;」は FILE 型のポインタ fp を宣言しているようですね。

厳密には違いますが（詳細は次章にて解説します)、考え方としてはあながち間違いではありません。

これは**ファイルポインタ**と呼ばれるものでファイルの入出力処理に必要です。また名前のとおり**変数はポインタとしなければなりません**。

```
12:          printf("ファイル名 : ");
13:          gets(fname);
```

ここはメッセージを表示して入力を受け付けています。

```
15:          /*ファイルを開く*/
16:          fp = fopen(fname, "r");
```

ここでは関数 fopen()を使ってファイルをオープンします。

また関数 fopen()の概要は以下のとおりです。

プロトタイプ	FILE *fopen(char *filename, char *mode)	
引数	char *filename	ファイル名
	char *mode	モード。主なモードは以下の通り "r" 読込み "w" 書込み "a" 追記 "r+" 読込みと書込み
返値	FILE *	オープンしたファイルのファイルポインタ。 オープンに失敗した場合は NULL を返す。
ヘッダファイル	stdio.h	

【解説】

ファイルを指定されたモードでオープンし、その結果をファイルポインタとして返す。ただし、ファイルが存在しない場合は NULL を返す。

【例】

```
fp = fopen("file.dat", "r");
```

「faile.dat」を読込みモードでオープンし、そのファイルポインタを fp に代入します。

よって、

```
15:         /*ファイルを開く*/
16:         fp = fopen(fname, "r");
```

の場合は読み取りモードでオープンしています。

それでは続きをたどってみましょう。

```
17:         if(fp == NULL){
18:                 printf("%s のオープンに失敗しました。¥n", fname);
19:                 exit(1);
20:         }
```

17 行目の if 文はファイルのオープンに失敗したら（ファイルポインタ fp の内容が NULL だったら）メッセージを表示して関数 exit()でプログラムを強制終了させます。

```
22:         /*一行ずつ読む*/
23:         buf = fgets(buf, LINESIZE, fp);
```

ここでは関数 fgets()を使って fp が指すファイルから一行だけ読込み、その内容を文字列 buf に書き込んでいます。

なお、関数 fgets()の概要は以下のとおりです。

プロトタイプ	char *fgets(char *str, int size, FILE *fp)	
引数	char *str	ファイルから一行だけ読み込んだ文字列
	int size	一行として読み込む最大の大きさ。これより長い文字列はその部分が切り捨てられる。
	FILE *fp	ファイルポインタ
返値	char *	読み込めた場合は str を返す。 ただしファイルの終端を読んだ場合は NULLL を返す。
ヘッダファイル	stdio.h	

【解説】

ファイルポインタが指すファイルから一行(ただし[size-1]文字まで)を読み、str に書き込む。

またファイルの終端を読んだ場合は NULL を返す。

【例】

```
p = fgets(s, 128, fp);
```

fp が指すファイルから一行（ただし 127 文字まで）を読み、その結果を p に、読み込んだ文字列を s に代入します。

gets()がキーボードから読むのに対し、この関数 fgets()はファイルから読むと考えれば理解しやすいでしょう。

gets()	キーボードから入力
fgets()	ファイルから入力

```
24:         while(buf != NULL){
25:                 printf("%s", buf);
26:                 buf = fgets(buf, LINESIZE, fp);
27:         }
```

24 行目では fgets()で読んだ文字列が NULL でない間（ファイルの終端まで読んでない間）25～26 行目の処理を繰り返します。

こうすることでファイルの最後まで一行ずつ読み、それを画面に表示することができます。

```
29:         /*ファイルを閉じる*/
30:         fclose(fp);
```

ここでは開いたファイルを閉じます。

開いたファイルは必ず閉じなければならないので気をつけましょう。

また関数 fclose()の概要は以下のとおりです。

プロトタイプ	int fclose(FILE *fp)	
引数	FILE *fp	ファイルポインタ
返値	int	クローズに成功した場合は 0、失敗した場合は EOF を返す。
ヘッダファイル	stdio.h	

【解説】

ファイルポインタが指すファイルをクローズします。

【例】

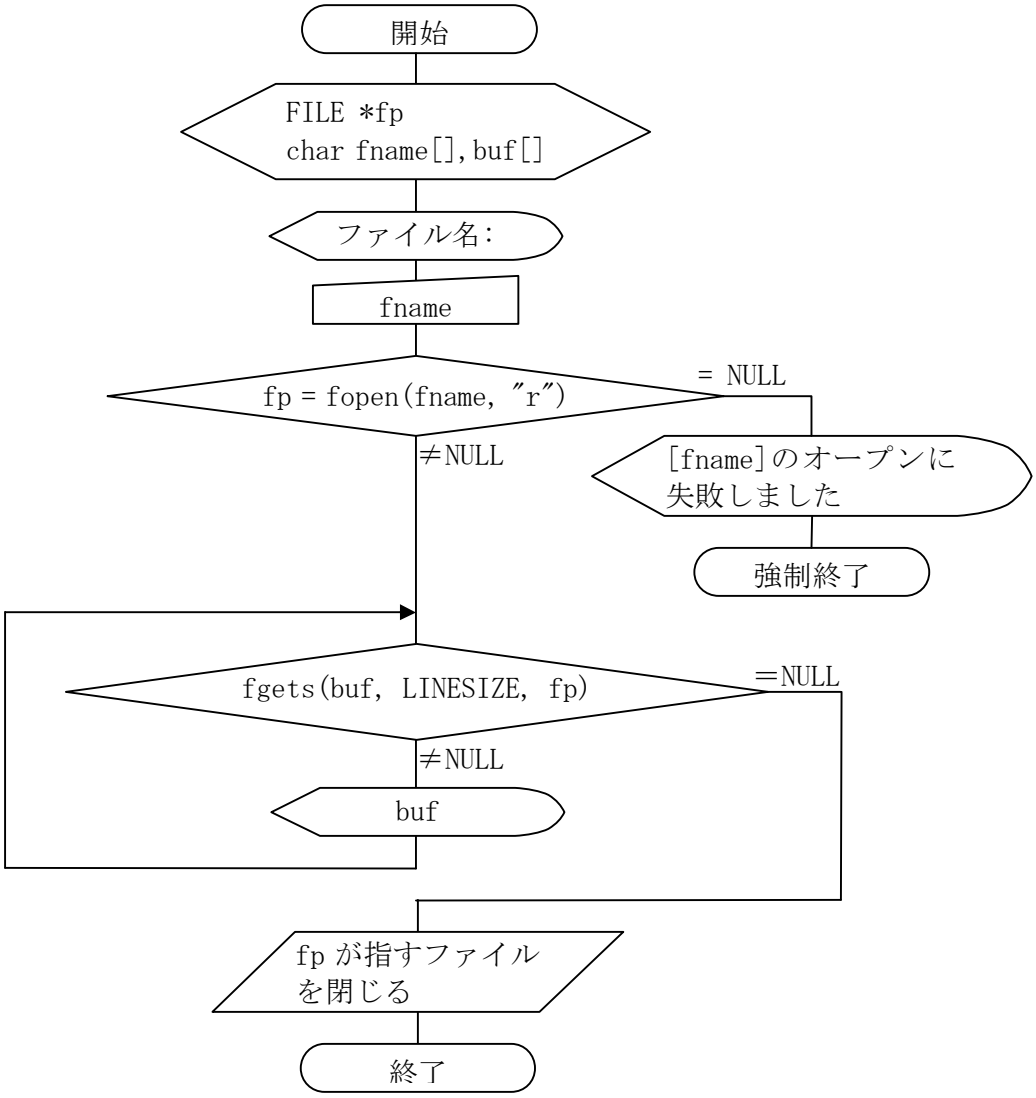
```
fclose(fp);
```

fp が指すファイルをクローズします。
(通例ではこのように返値を確認しません)

関数 fclose()の引数はファイル名ではありませんので注意しましょう。

1.2. より簡潔にする

fileread1.cpp のフローチャートとソースコードを簡潔にしてみました。



fileread2.cpp

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define NAMESIZE 256
 5:  #define LINESIZE 1024
 6:
 7:  main()
 8:  {
 9:          FILE    *fp;
10:          char    fname[NAMESIZE], buf[LINESIZE];
11:
12:          printf("ファイル名 : ");
13:          gets(fname);
14:
15:          /*ファイルを開く*/
16:          if((fp = fopen(fname, "r")) == NULL){
17:                  printf("%s のオープンに失敗しました。¥n", fname);
18:                  exit(1);
19:          }
20:
21:          /*一行ずつ読む*/
22:          while((fgets(buf, LINESIZE, fp)) != NULL){
23:                  printf("%s", buf);
24:          }
25:
26:          /*ファイルを閉じる*/
27:          fclose(fp);
28:  }
```

このように if や while 文の条件式に関数を記入することが可能です。

1.3. ファイルに書き込む

続いて、ファイルに出力するプログラムを作りましょう。

file2.cpp

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define SIZE    128
 5:
 6:  double doubleinput(char *);
 7:
 8:  main()
 9:  {
10:          FILE    *fp;
11:          char    fname[SIZE], buf;
12:          double  a, b;
13:
14:          printf("ファイル名 : ");
15:          gets(fname);
16:
17:          a = doubleinput("a = ");
18:          b = doubleinput("b = ");
19:
20:          if((fp = fopen(fname, "w")) == NULL){
21:                  printf("%s のオープンに失敗しました。¥n", fname);
22:                  exit(1);
23:          }
24:
25:          fprintf(fp, "a + b = %g¥n", a + b);
26:          fprintf(fp, "a - b = %g¥n", a - b);
27:          fprintf(fp, "a * b = %g¥n", a * b);
28:          fprintf(fp, "a / b = %g¥n", a / b);
29:
30:          fclose(fp);
31:  }
32:
33:  double doubleinput(char *msg)
34:  {
35:          double  val;
36:
```

```
37:             printf(msg);
38:             scanf("%lf", &val);
39:
40:             return  val;
41:     }
```

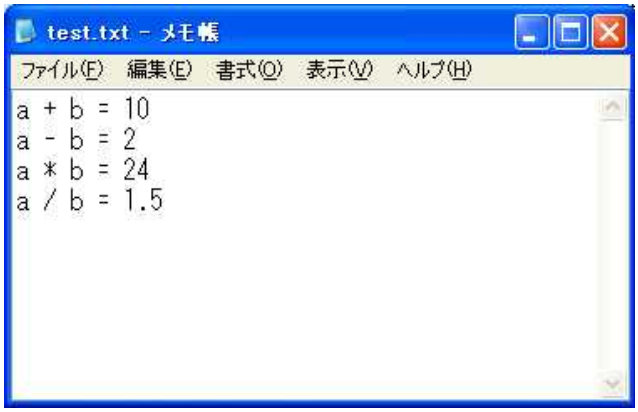
実行例

ファイル名 ： test.txt

a = 6

b = 4

実行後、指定したファイル名（上記の例では「test.txt」）が作成されます。
それをメモ帳などで開いてみましょう。



ファイルに書き込まれているのがわかります。

それでは重要な部分の解説をしましょう。

```
20:             if((fp = fopen(fname, "w")) == NULL){
21:                     printf("%s のオープンに失敗しました。¥n", fname);
22:                     exit(1);
23:             }
```

今回はファイルへの書込みを行うのでモードを"w"としています。

```
25:         fprintf(fp, "a + b = %g¥n", a + b);
26:         fprintf(fp, "a - b = %g¥n", a - b);
27:         fprintf(fp, "a * b = %g¥n", a * b);
28:         fprintf(fp, "a / b = %g¥n", a / b);
```

なんとなく関数 printf()や関数 sprintf に似ていますね。
この関数 fprintf()は画面や文字配列ではなくファイルに出力します。

printf()	画面に出力
sprintf()	文字列に出力
fprintf()	ファイルに出力

関数 fprintf()の概要は以下のとおりです。

プロトタイプ	int fprintf(FILE *fp, char *format, ...)	
引数	FILE *fp	ファイルポインタ
	char *format	表示する文字列 %d など変換仕様の記述が可能
	...	format に記述された変換仕様に対応する式
返値	int	出力する半角文字数を返す ただしエラーが発生すると EOF を返す
ヘッダファイル	stdio.h	

【解説】

format に記述された内容に従って文字列を fp が指すファイルに出力する。

【例】

```
fprintf(fp, "a = %d¥n", a);
```

生成した文字列を fp が指すファイルに書き込みます。
(通例ではこのように返値を確認しません)

2.　CSV ファイルにアクセスする

CSV ファイルとはカンマで区切られたデータファイルのことです。
この形式のファイルは Microsoft Excel など表計算ソフトで開くことができますので自作した C プログラムと表計算ソフトを連携させることも可能です。

※ 本書では Microsoft Excel を例に解説しています。他の表計算ソフトを使用する場合は用語を適宜読み替えて下さい。

2.1. CSV ファイルを作成する

以下の CSV ファイルの集計プログラムを作ってみましょう。

- 一行が生徒一人の国語と数学の点数
- 表は点数のみで、見出しは無い
- 生徒の人数は不定とする

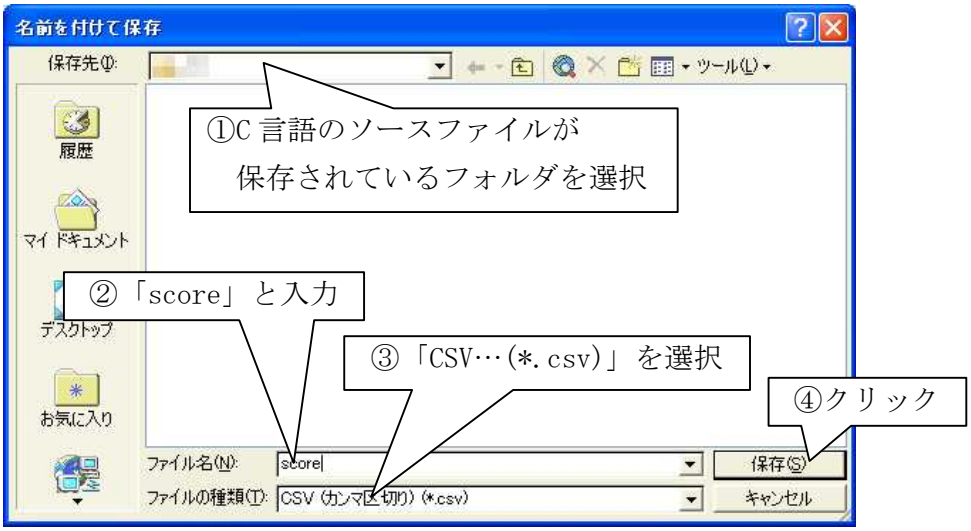
	A	B
1	100	80
2	60	50
3	70	80
4	90	60
5	80	80
6	50	70
7	90	100
8	80	90
9	70	90
10	60	70

Microsoft Excel を起動し左図のような表を作成します。

メニュー「ファイル(F)」-「名前をつけて保存(A)...」をクリックします。
「名前をつけて保存」ダイアログボックスが表示されるので、

- **「保存先(I)」**　　**ソースファイルが保存されているフォルダ**
- **「ファイル名(N)」**　　score
- **「ファイルの種類(T)」**　CSV（カンマ区切り）(*.CSV)

として「保存(S)」ボタンをクリックします。



保存するときいくつかの警告メッセージが表示されることがあります。
これは特に問題はありませんので保存を実行させてください。

2.2. CSV ファイルを読み、画面に表示する

まずは CSV ファイルを読込むプログラムを作ってみましょう。

syukei1.cpp

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define LINESIZE 128
 5:
 6:  main()
 7:  {
 8:          FILE    *in;
 9:          char    str[LINESIZE];
10:
11:          if((in = fopen("score.csv", "r")) == NULL){
12:                  printf("score.csv のオープンに失敗しました。¥n");
13:                  exit(1);
14:          }
15:
16:          while(fgets(str, LINESIZE, in) != NULL){
17:                  printf("%s¥n", str);
18:          }
19:
20:          fclose(in);
21:  }
```

実行例

```
100, 80
60, 50
70, 80
90, 60
80, 80
50, 70
90, 100
80, 90
70, 90
60, 70
```

このように CSV ファイルの内容がそのまま表示されます。

2.3. 国語と数学の点数を抜き出す

とりあえず読込みには成功したので、今度は関数 sscanf()を使って数値を抜き出し個別に表示してみましょう。

syukei2.cpp

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define LINESIZE 128
 5:
 6:  main()
 7:  {
 8:          FILE    *in;
 9:          char    str[LINESIZE];
10:          int     kokugo, sugaku;
11:
12:          if((in = fopen("score.csv", "r")) == NULL){
13:                  printf("score.csv のオープンに失敗しました。¥n");
14:                  exit(1);
15:          }
16:
17:          while(fgets(str, LINESIZE, in) != NULL){
18:                  sscanf(str, "%d,%d", &kokugo, &sugaku);
19:                  printf("国語:%3d, 数学:%3d¥n", kokugo, sugaku);
20:          }
21:
22:          fclose(in);
23:  }
```

実行例

```
国語:100,  数学:  80
国語:  60,  数学:  50
国語:  70,  数学:  80
国語:  90,  数学:  60
国語:  80,  数学:  80
国語:  50,  数学:  70
国語:  90,  数学:100
国語:  80,  数学:  90
国語:  70,  数学:  90
国語:  60,  数学:  70
```

どうやらうまく取り出せたようです。

2.4. 国語と数学の平均点を算出し表示する

続いて、取り出した点数から平均点を算出しましょう。

syukei3.cpp

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define LINESIZE 128
 5:
 6:  main()
 7:  {
 8:          FILE    *in;
 9:          char    str[LINESIZE];
10:          int     kokugo, sugaku;
11:          double  heikin;
12:
13:          if((in = fopen("score.csv", "r")) == NULL){
14:                  printf("score.csv のオープンに失敗しました。¥n");
15:                  exit(1);
16:          }
17:
18:          while(fgets(str, LINESIZE, in) != NULL){
19:                  sscanf(str, "%d,%d", &kokugo, &sugaku);
20:                  heikin = (kokugo + sugaku) / 2.0;
21:                  printf("国語:%3d, 数学:%3d, 平均:%5.1f¥n", kokugo, sugaku, heikin);
22:          }
23:
24:          fclose(in);
25:  }
```

実行例

```
国語:100, 数学: 80, 平均: 90.0
国語: 60, 数学: 50, 平均: 55.0
国語: 70, 数学: 80, 平均: 75.0
国語: 90, 数学: 60, 平均: 75.0
国語: 80, 数学: 80, 平均: 80.0
（以下省略）
```

2.5. CSV 形式に整形する

今度は国語、数学、平均点を CSV 形式に整形しましょう。
ファイルへの書込みまであと一歩です。

syukei4.cpp

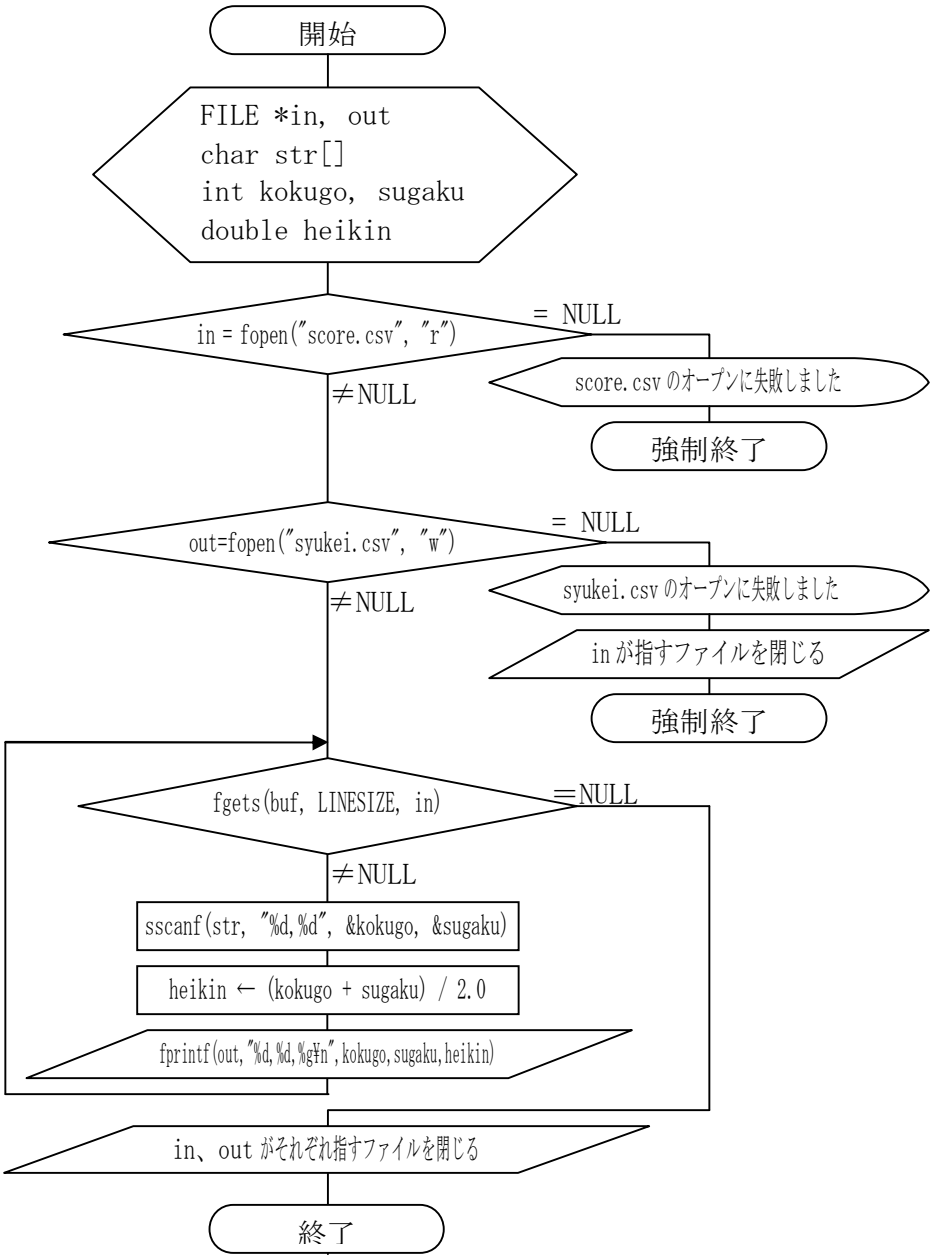
```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define LINESIZE 128
 5:
 6:  main()
 7:  {
 8:          FILE    *in;
 9:          char    str[LINESIZE];
10:          int     kokugo, sugaku;
11:          double  heikin;
12:
13:          if((in = fopen("score.csv", "r")) == NULL){
14:                  printf("score.csv のオープンに失敗しました。¥n");
15:                  exit(1);
16:          }
17:
18:          while(fgets(str, LINESIZE, in) != NULL){
19:                  sscanf(str, "%d,%d", &kokugo, &sugaku);
20:                  heikin = (kokugo + sugaku) / 2.0;
21:                  printf("%d,%d,%g¥n", kokugo, sugaku, heikin);
22:          }
23:
24:          fclose(in);
25:  }
```

実行例

```
100,80,90
60,50,55
70,80,75
90,60,75
80,80,80
（以下省略）
```

2.6. 「syukei.csv」に出力する

いよいよファイルに書込みます。
そのフローチャートとソースコードは以下の通りです。



syukei5.cpp

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define LINESIZE 128
 5:
 6:  main()
 7:  {
 8:          FILE    *in, *out;
 9:          char    str[LINESIZE];
10:          int     kokugo, sugaku;
11:          double  heikin;
12:
13:          if((in = fopen("score.csv", "r")) == NULL){
14:                  printf("score.csv のオープンに失敗しました。¥n");
15:                  exit(1);
16:          }
17:          if((out = fopen("syukei.csv", "w")) == NULL){
18:                  printf("syukei.csv のオープンに失敗しました。¥n");
19:                  fclose(in);
20:                  exit(1);
21:          }
22:
23:          while(fgets(str, LINESIZE, in) != NULL){
24:                  sscanf(str, "%d,%d", &kokugo, &sugaku);
25:                  heikin = (kokugo + sugaku) / 2.0;
26:                  fprintf(out, "%d,%d,%g¥n", kokugo, sugaku, heikin);
27:          }
28:
29:          fclose(in);
30:          fclose(out);
31:  }
```

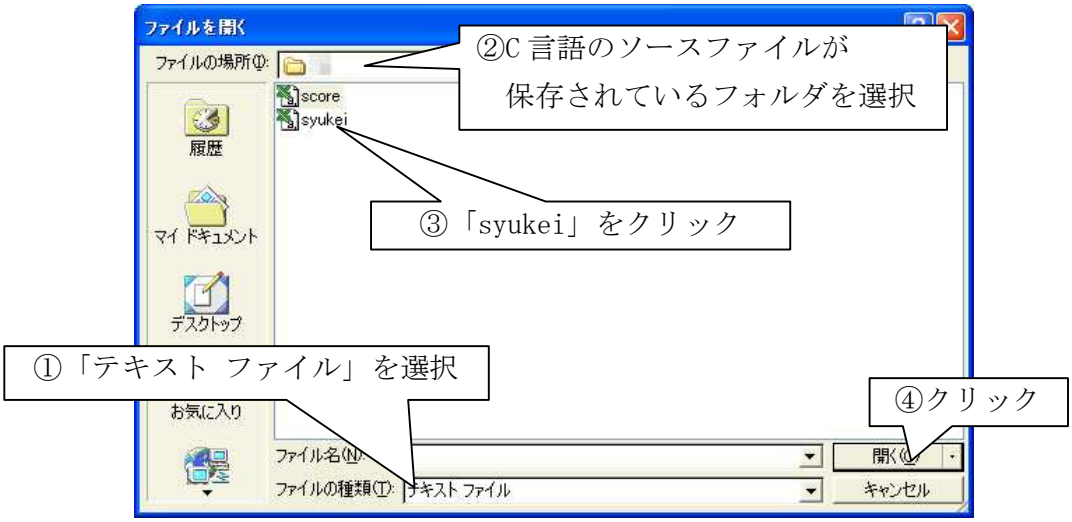
当然ですが、実行しても画面には何も表示されません。

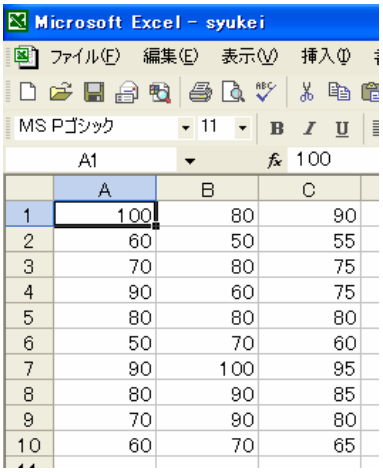
Microsoft Excel を起動しメニュー「ファイル(F)」-「開く(O)」をクリックします。

このとき「ファイルを開く」ダイアログボックスが表示されるので、

- 「ファイルの種類(T)」　テキストファイル
- 「ファイルの場所(I)」　ソースファイルが保存されているフォルダ
- 「ファイル名(N)」　syukei

として「開く(O)」ボタンをクリックします。





	A	B	C
1	100	80	90
2	60	50	55
3	70	80	75
4	90	60	75
5	80	80	80
6	50	70	60
7	90	100	95
8	80	90	85
9	70	90	80
10	60	70	65

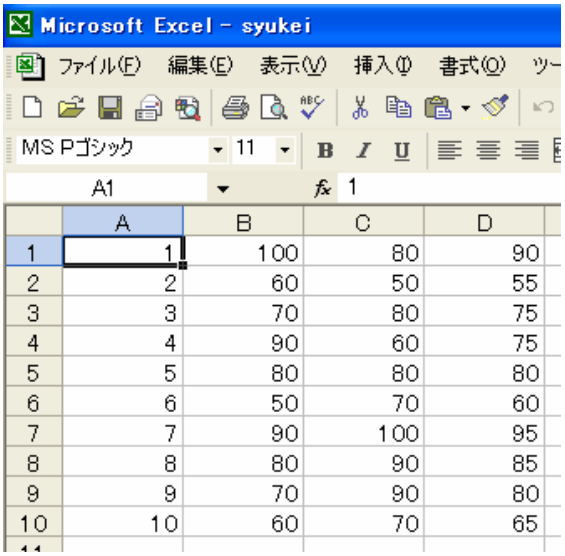
集計された成績ファイルを読むことができるようになりました。

2.7. 連番や見出しをつける

「syukei.csv」を出力する際に、生徒番号を振ります。
なお、生徒番号は１からの連番とします。

syukei6.cpp

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define LINESIZE 128
 5:
 6:  main()
 7:  {
 8:          FILE    *in, *out;
 9:          char    str[LINESIZE];
10:          int     kokugo, sugaku;
11:          double  heikin;
12:          int     n;
13:
14:          if((in = fopen("score.csv", "r")) == NULL){
15:                  printf("score.csv のオープンに失敗しました。¥n");
16:                  exit(1);
17:          }
18:          if((out = fopen("syukei.csv", "w")) == NULL){
19:                  printf("syukei.csv のオープンに失敗しました。¥n");
20:                  fclose(in);
21:                  exit(1);
22:          }
23:
24:          for(n = 1; fgets(str, LINESIZE, in) != NULL; n++){
25:                  sscanf(str, "%d,%d", &kokugo, &sugaku);
26:                  heikin = (kokugo + sugaku) / 2.0;
27:                  fprintf(out, "%d,%d,%d,%g¥n", n, kokugo, sugaku, heikin);
28:          }
29:
30:          fclose(in);
31:          fclose(out);
     }
```



	A	B	C	D
1	1	100	80	90
2	2	60	50	55
3	3	70	80	75
4	4	90	60	75
5	5	80	80	80
6	6	50	70	60
7	7	90	100	95
8	8	80	90	85
9	9	70	90	80
10	10	60	70	65

一列目（A 列）に連番が振られました。

数字だけの表ではわかりにくいので、一行目に見出しをつけるようにしましょう。

syukei7.cpp

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define  LINESIZE 128
 5:
 6:  main()
 7:  {
 8:          FILE     *in, *out;
 9:          char     str[LINESIZE];
10:          int      kokugo, sugaku;
11:          double   heikin;
12:          int      n;
13:
14:          if((in = fopen("score.csv", "r")) == NULL){
15:                  printf("score.csv のオープンに失敗しました。¥n");
16:                  exit(1);
17:          }
18:          if((out = fopen("syukei.csv", "w")) == NULL){
19:                  printf("syukei.csv のオープンに失敗しました。¥n");
20:                  fclose(in);
```

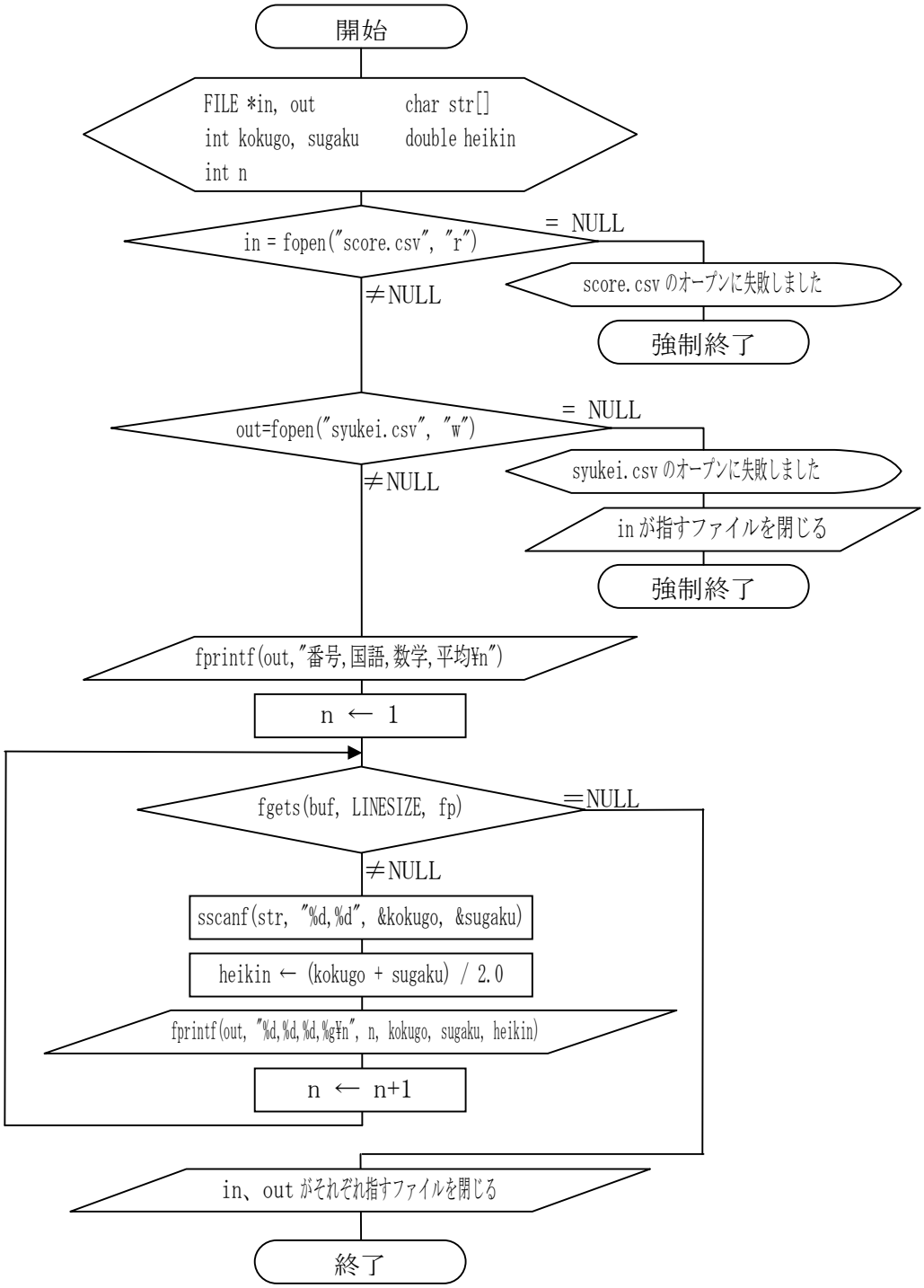
```
21:                     exit(1);
22:             }
23:
24:             fprintf(out,"番号,国語,数学,平均¥n");
25:             for(n = 1; fgets(str, LINESIZE, in) != NULL; n++){
26:                     sscanf(str, "%d,%d", &kokugo, &sugaku);
27:                     heikin = (kokugo + sugaku) / 2.0;
28:                     fprintf(out, "%d,%d,%d,%g¥n", n, kokugo, sugaku, heikin);
29:             }
30:
31:             fclose(in);
32:             fclose(out);
33:     }
```

	A	B	C	D
1	番号	国語	数学	平均
2	1	100	80	90
3	2	60	50	55
4	3	70	80	75
5	4	90	60	75
6	5	80	80	80
7	6	50	70	60
8	7	90	100	95
9	8	80	90	85
10	9	70	90	80
11	10	60	70	65

1行目に見出しが入りました。

これで幾分か見やすい集計表になりましたね。

なお、syukei7 のフローチャートは以下の通りです。参考にしてください。



3. 演習問題

14-1. 「syukei7.cpp」を改造して以下のようなCSVファイルを出力するプログラム「ensyu14-1.cpp」を作りなさい。

	A	B	C	D	E
1	番号	国語	数学	合計	平均
2	1	100	80	180	90
3	2	60	50	110	55
4	3	70	80	150	75
5	4	90	60	150	75
6	5	80	80	160	80
7	6	50	70	120	60
8	7	90	100	190	95
9	8	80	90	170	85
10	9	70	90	160	80
11	10	60	70	130	65

14-2. 「ensyu14-1.cpp」を参考に以下のようなCSVファイルを出力するプログラム「ensyu14-2.cpp」を作りなさい。

	A	B	C	D	E
1	国語と数学の試験結果				
2	番号	国語	数学	合計	平均
3	1	100	80	180	90
4	2	60	50	110	55
5	3	70	80	150	75
6	4	90	60	150	75
7	5	80	80	160	80
8	6	50	70	120	60
9	7	90	100	190	95
10	8	80	90	170	85
11	9	70	90	160	80
12	10	60	70	130	65