

2021

計算機プログラミング演習

第10回 ファイルの入出力(1)

第10回の目標

- ファイルの入出力方法を理解する
- ファイルの種類によって方法が変わる
 - テキストファイル
 - バイナリファイル
- それぞれの方法の利点と欠点を理解し、使いこなせるように

ファイルの入出力

- ファイルの種類

本日の演習はこちら

- テキストファイル

- 文字列(テキスト)を収めたファイル
 - エディタで開いて読むことができるファイル

- バイナリファイル

次回の演習で実施

- 変数の内容を(文字に変換せずに)直接ファイルに書き込む方法
 - 数値を変数内のデータ形式のまま保存
 - 整数(int) 4 byte, 倍精度実数(double) 8 byteでファイルに格納(区切り記号、改行なし)
 - 実行ファイルやデータファイル、音声や画像など

テキストファイルのメリット・デメリット

- メリット

- テキストエディタ等で内容を見ることができる
- 複数のプログラム間のデータの共有が容易
- 異種コンピュータ、OS 間でのデータの共有可

- デメリット

- ファイルサイズが大きくなる
- 書き出し・読み込みプログラムが煩雑になる
 - 変数とその種類によってプログラムが変わる
 - 配列は繰り返し(for など)を用いて出力

バイナリーファイルのメリット・デメリット

- メリット

- 配列や構造体などの多数(まとまった)変数の保存が容易
- 数値のファイルサイズが小さくなる

- デメリット

- 内容を見ても判別が困難
 - データ保存時のプログラムに大きく依存
- 異種コンピュータ、OS 間でのデータの共有に難
 - バイト並び(エンディアン問題)

ファイルの入出力方法(1)

- 共通の使い方

- `fopen()` でファイルを「開く」
 - ファイル名 `"filename"`
 - 書き込み (`w`) か 読み込み (`r`) か
 - テキストファイル か バイナリファイル (`b`) か
 - ファイルポインタを取得する
- `fclose()` でファイルを「閉じる」
 - `fopen()` で開いたファイルは必ず閉じる(重要)

```
FILE    * fp ;                // ファイルポインタの宣言
fp = fopen("filename", "r" ) ; // ファイルを開く
      :
fclose( fp ) ;                // ファイルを閉じる
```

ファイルの入出力方法(2)

- テキストファイル

- `fopen()` で開いたファイルに対して

- `fprintf()` でテキストデータを書き出す

```
fprintf( fp, "%d\n", i ) ;
```

- `fscanf()` でテキストデータを読み込む

```
fscanf( fp, "%d", &i ) ;
```

教科書 p.351 のプログラム(1)

```
/*    file_test1.c    */

#define    _CRT_SECURE_NO_WARNINGS    1

#include    <stdio.h>

int    main(void) {

    FILE    * file;        // ファイルポインタ

    file = fopen( "test.txt" , "w"); // ファイルを開く
                                    // "w" で書き込み指定

    fclose(file);        // ファイルを閉じる

    return 0;
}
```

fopen() で開いたファイルは**必ず** fclose()
で閉じる！（エラーは出ないが、危険）

実行結果

実行前

実行

実行後

```
コマンドプロンプト
Microsoft Windows [Version 10.0.19042.1055]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Nakano>cd C:\Users\Nakano\source\repos\file_test1\Debug

C:\Users\Nakano\source\repos\file_test1\Debug>dir
ドライブ C のボリューム ラベルは Windows です
ボリューム シリアル番号は 3427-890F です

C:\Users\Nakano\source\repos\file_test1\Debug のディレクトリ
2021/06/15  11:14    <DIR>          .
2021/06/15  11:14    <DIR>          ..
                39,936 file_test1.exe
                389,412 file_test1.ilc
                438,272 file_test1.pdb
                3 個のファイル             867,620 バイト
                2 個のディレクトリ  723,820,519,424 バイトの空き領域

C:\Users\Nakano\source\repos\file_test1\Debug>file_test1

C:\Users\Nakano\source\repos\file_test1\Debug>dir
ドライブ C のボリューム ラベルは Windows です
ボリューム シリアル番号は 3427-890F です

C:\Users\Nakano\source\repos\file_test1\Debug のディレクトリ
2021/06/15  11:16    <DIR>          .
2021/06/15  11:16    <DIR>          ..
                39,936 file_test1.exe
                389,412 file_test1.ilc
                438,272 file_test1.pdb
                0 test.txt
                4 個のファイル             867,620 バイト
                2 個のディレクトリ  723,819,995,136 バイトの空き領域

C:\Users\Nakano\source\repos\file_test1\Debug>
```

実行ファイル **file_test1.exe** と同じディレクトリに
test.txt というサイズ 0 のファイルが作成されている。

p.351 のプログラム(2)エラー処理追加

```
/*      file_test1.c      */
#define  _CRT_SECURE_NO_WARNINGS 1
#include <stdio.h>

int  main(void) {

    FILE  * file;      // ファイルポインタ

    file = fopen("test.txt", "w"); // ファイルを開く

    if (file == NULL) {
        printf("ファイルをオープンできませんでした。¥n");
        return 1;      // エラー(1)で終了
    }

    fclose(file);      // ファイルを閉じる

    return 0;          // 正常(0)終了
}
```

NULLって何？

- NULL(ヌル)とは「何も示さないもの」を指す
- NULLは数値のゼロや空文字とは違う

ゼロは数値として、空文字(""となる長さゼロの文字列)は文字としての意味を持っているが、
NULLは何も意味を持たない

具体的にいうと、NULLとはデータではなくアドレスの状態であるが、ポインタの話のときに再度説明する

fprintf() で文字を書いてみる

```
/*      file_test1.c      */
#define _CRT_SECURE_NO_WARNINGS 1
#include <stdio.h>

int main(void) {

    FILE * file;          // ファイルポインタ

    file = fopen("test.txt", "w"); // ファイルを開く
    if (file == NULL) {
        printf("ファイルをオープンできませんでした。¥n");
        return 1;
    }

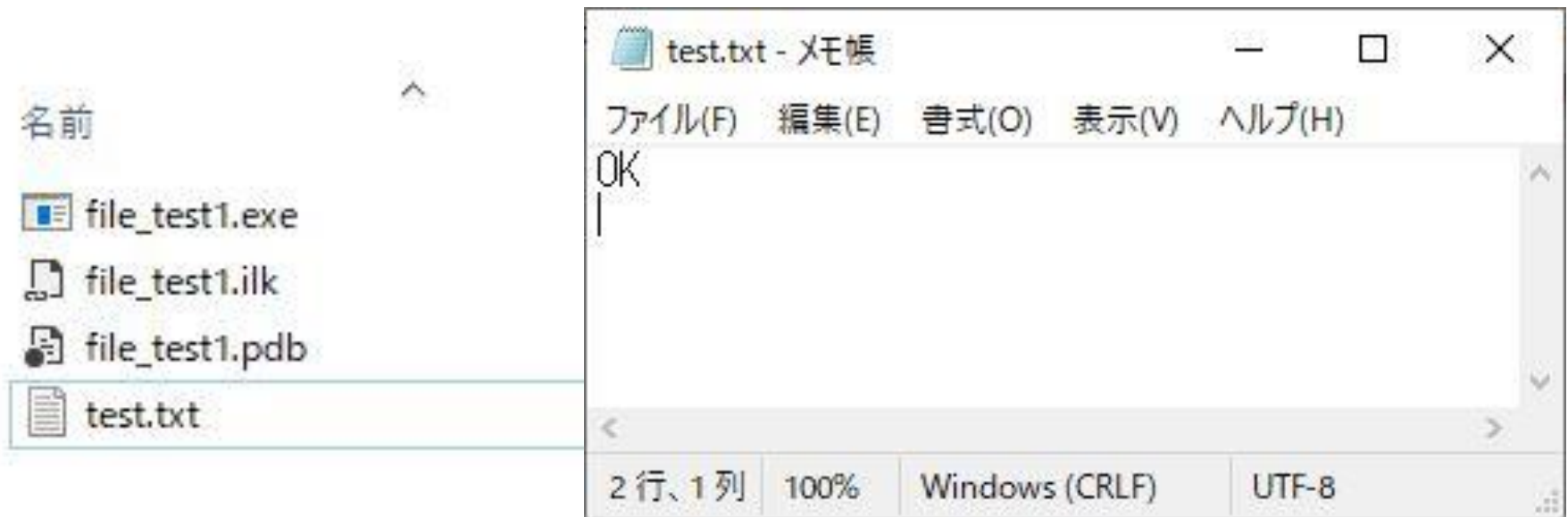
    fprintf( file, "OK¥n" ) ;

    fclose( file );

    return 0;
}
```

生成されたtest.txt ファイルはどこにある？

- ビルドによって作成された実行プログラムと同じディレクトリにできる。



test.txt をダブルクリックして開く

csv ファイルを作る

- 数値と数値の間を“, “ (カンマ) または“¥t” (タブ) および “¥n” (改行) で区切ったテキストファイルの形式。
- Excel で読み込める。
- 計測器からのデータ引き渡しで標準的な形式。

CSV ファイルの出力 (1)

```
/*      file_test2.c      */

#define _CRT_SECURE_NO_WARNINGS 1
#include <stdio.h>
#include <math.h>

int main(void) {

    FILE          *file;
    int            i ;
    double         th;

    file = fopen("test2.csv", "w");
    if (file == NULL) {
        printf("ファイルをオープンできません。      ¥n");
        return 1;
    }
}
```

データ区切りが ", " (カンマ) の CSV ファイルにする

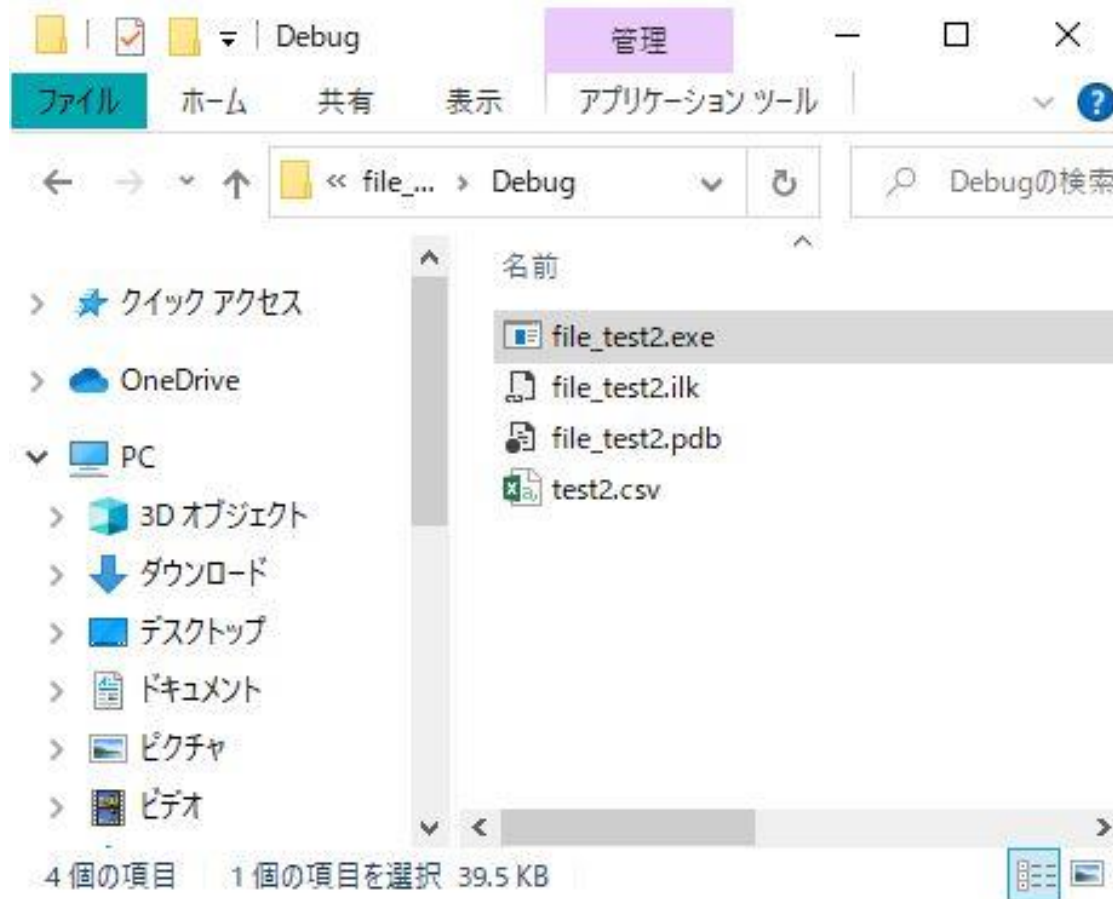
CSV ファイルの出力 (2)

```
for (i = 0; i <= 360; i += 10) {  
    th = 3.141592653 / 180 * i;  
  
    fprintf(file, "%d, %f\\n", i,  
            sin(th));  
}  
  
fclose(file);  
  
return 0;  
}
```

データ区切りが "," (カンマ)

```
C:\Users\Nakano\source\repos\file_test2\Debug>file_test2  
C:\Users\Nakano\source\repos\file_test2\Debug>
```


実行結果



- Debug フォルダ内にtest2.csv ファイルができる。
 - Excel の文書アイコンになっている。

Excel 起動

- データがExcelに読み込まれている。

test2.csv - Excel

ファイル ホーム 挿入 ページレイアウト 数式 データ

クリップボード フォント 配置 数値 スタイル

条件付き書式 テーブルとして書式設定 セルのスタイル

A1 : fx 0

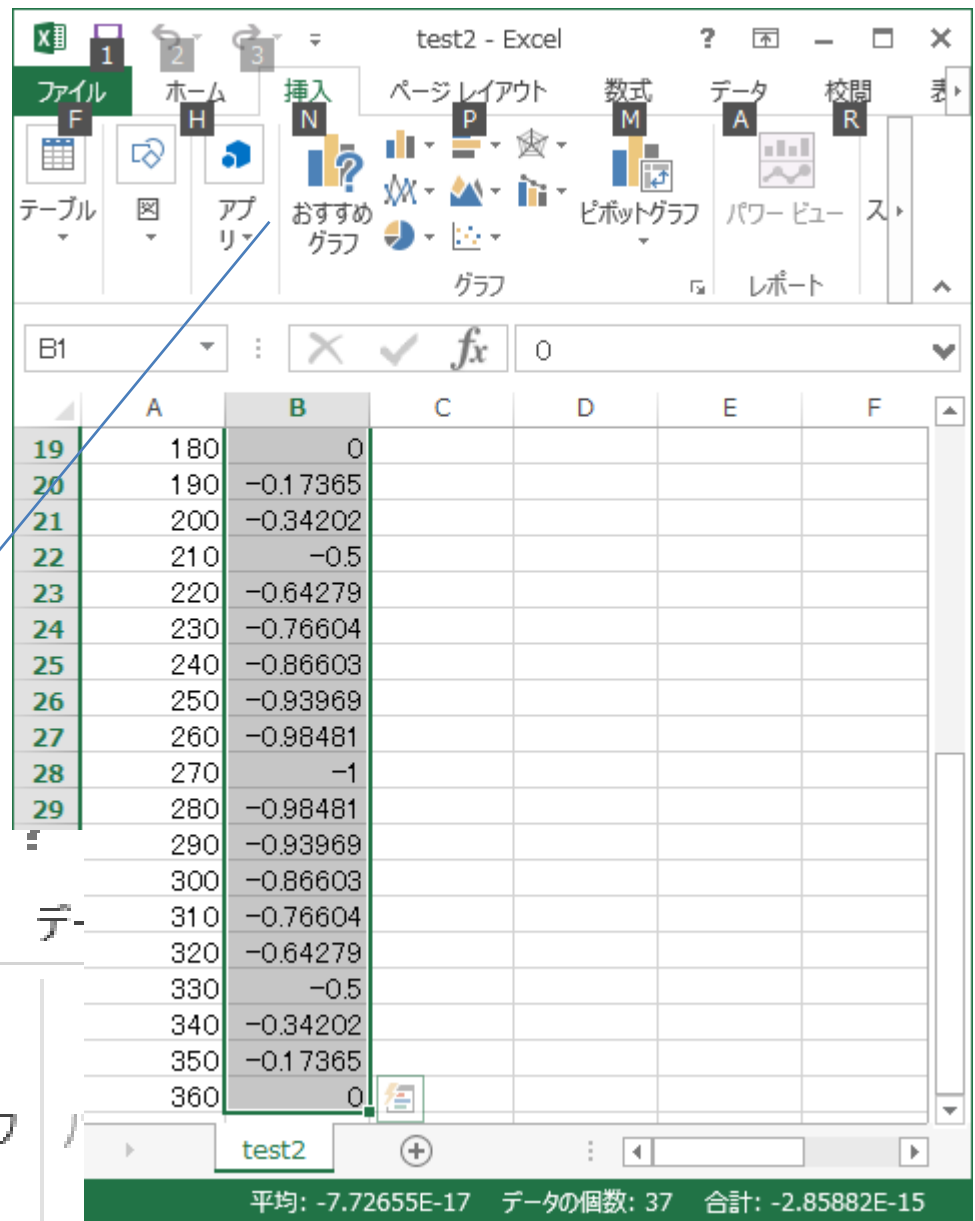
| | A | B | C | D | E |
|----|-----|----------|---|---|---|
| 1 | 0 | 0 | | | |
| 2 | 10 | 0.173648 | | | |
| 3 | 20 | 0.34202 | | | |
| 4 | 30 | 0.5 | | | |
| 5 | 40 | 0.642788 | | | |
| 6 | 50 | 0.766044 | | | |
| 7 | 60 | 0.866025 | | | |
| 8 | 70 | 0.939693 | | | |
| 9 | 80 | 0.984808 | | | |
| 10 | 90 | 1 | | | |
| 11 | 100 | 0.984808 | | | |
| 12 | 110 | 0.939693 | | | |
| 13 | 120 | 0.866025 | | | |
| 14 | 130 | 0.766044 | | | |
| 15 | 140 | 0.642788 | | | |
| 16 | 150 | 0.5 | | | |

test2

準備完了 100%

Excel でグラフ

- B1 - B37 を選択して、「挿入」->折れ線グラフ->グラフ形式の選択



Excel でグラフ

B列を選択 -> 挿入タブ -> 折れ線グラフ



テキストファイル読み込みプログラム (1)

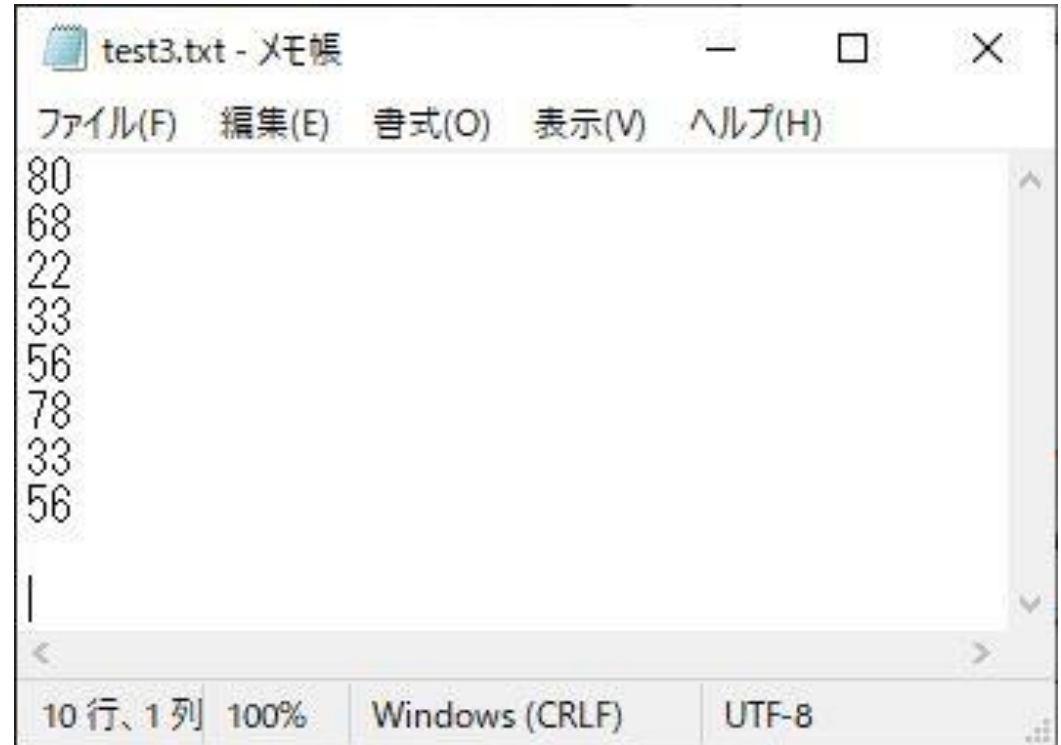
テキストファイル
の準備

右クリック

->[新規作成(x)]

->[テキスト ド
キュメント]

->「test3.txt」



ビルド後にDebugフォルダ内に移動させる

テキストファイル読み込みプログラム(2)

```
/*    file_test3.c    */

#define _CRT_SECURE_NO_WARNINGS 1
#include <stdio.h>
#define NUM 8    //記号定数の定義

int main(void) {

    FILE        *fp;
    int          test[NUM];
    int          max, min;
    int          i, j;
```

テキストファイル読み込みプログラム(3)

```
fp = fopen("test3.txt", "r");

if (fp == NULL) {
    printf("ファイルをオープンできませんでした。
¥n");
    return 1;
}

for (i = 0; i < NUM; i++) {
    fscanf(fp, "%d", &test[i]);
}
```

テキストファイル読み込みプログラム(4)

```
max = test[0];
min = test[0];

for (j = 0; j < NUM; j++) {
    if (max < test[j]) {
        max = test[j];
    }
    if (min > test[j]) {
        min = test[j];
    }

    printf("No. %-5d%d¥n", j+1, test[j]);

}
```


テキストファイル読み込みプログラム(5)

```
printf("最高点は%dです。¥n", max);  
printf("最低点は%dです。¥n", min);
```

```
fclose(fp);
```

```
return 0;
```

```
}
```

実行結果

```
C:\Users\Nakano\source\repos\file_test3\Debug>file_test3  
No. 1      80  
No. 2      68  
No. 3      22  
No. 4      33  
No. 5      56  
No. 6      78  
No. 7      33  
No. 8      56  
最高点は80です。  
最低点は22です。
```

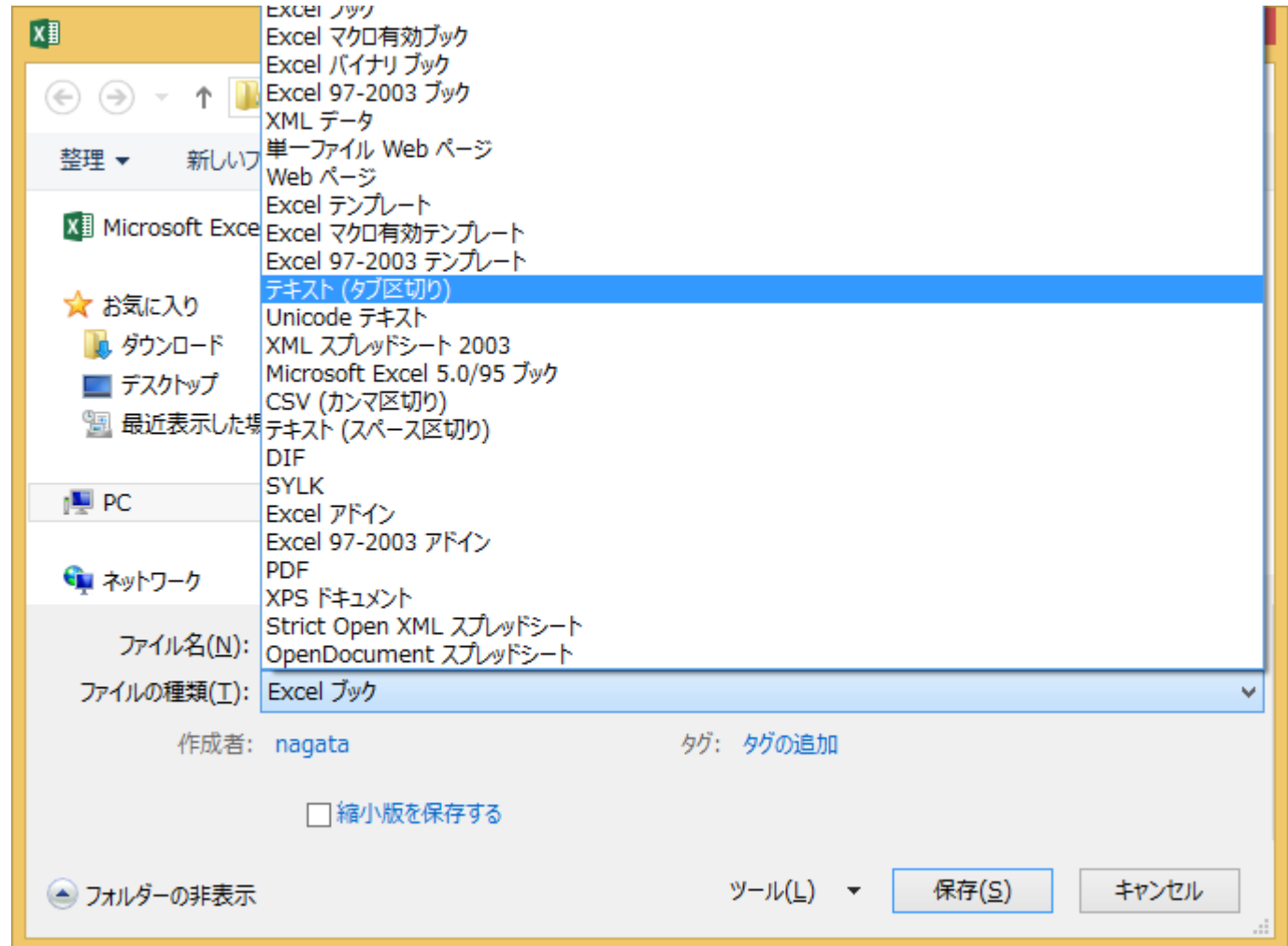
Excel からのファイル入力(1)

- Excel で
ファイルの
準備



Excel からのファイル入力(2)

- 入力が終わったら「テキスト(タブ区切り)」で保存
- test4.txt



ビルド後にDebugフォルダ内に移動させる

Excel からのファイル入力(3)

```
/*    file_test4.c ファイル入出力 */

#define _CRT_SECURE_NO_WARNINGS 1
#include <stdio.h>

int  main(void) {

    FILE        *fp;
    int          mark[8][3];
    int          max[3], min[3];
    int          i, j;
    char         sub[3][5] =
        { "国語", "数学", "英語" };
```

Excel からのファイル入力(4)

```
fp = fopen("test4.txt", "r");
```

```
if (fp == NULL) {  
    printf("ファイルをオープンできませんでした。¥n");  
    return 1;  
}
```

データ区切りが " " (スペース)であることに注意。

```
for (i = 0; i < 8; i++) {  
    fscanf(fp, "%d %d %d", &mark[i][0],  
            &mark[i][1],  
            &mark[i][2]);  
}
```

カンマ区切り (csv ファイルなど) の場合、"%d, %d, %d" を使う。 (scanf () で説明したのと同様)

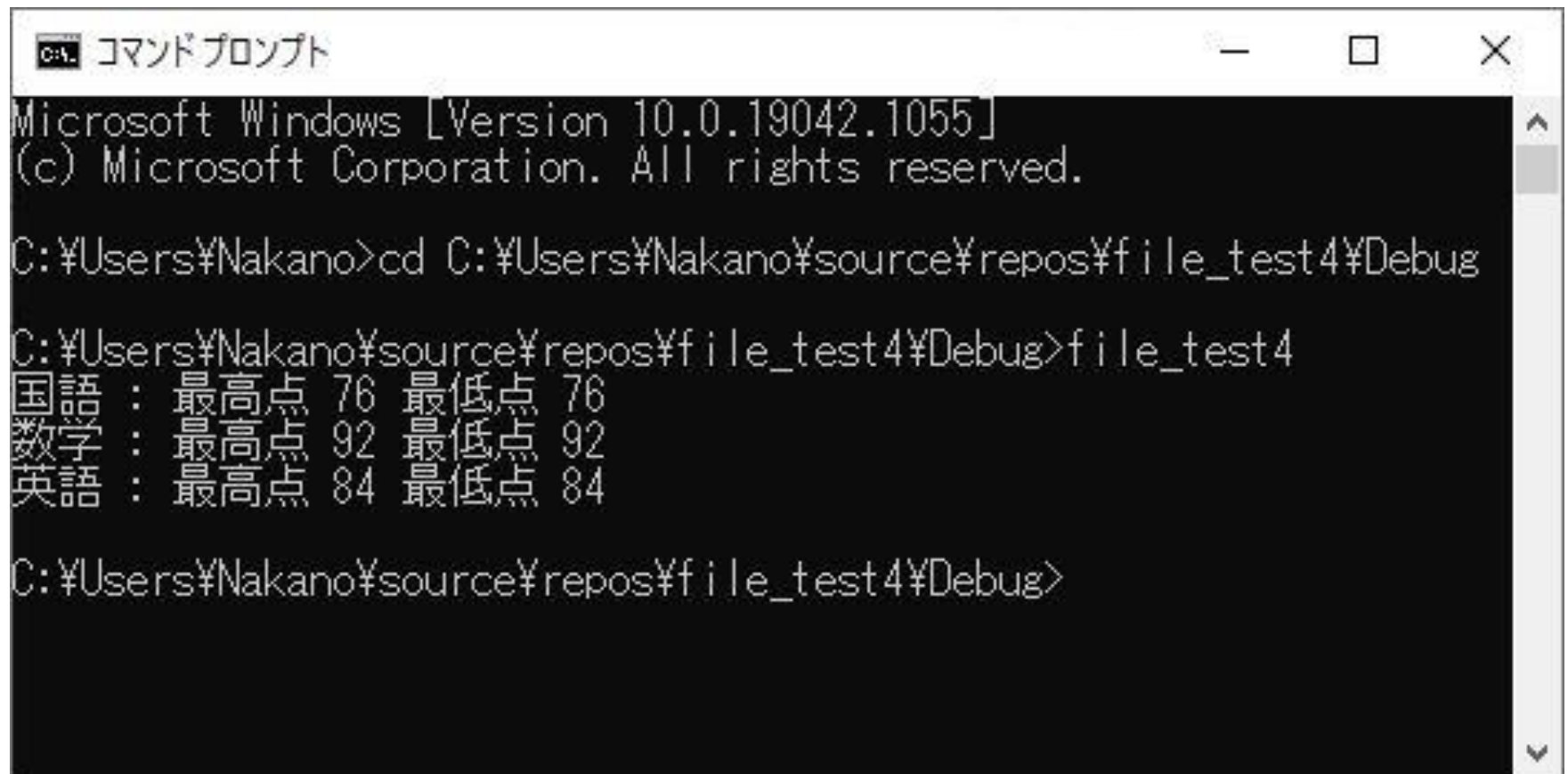
Excel からのファイル入力(5)

```
for (i = 0; i < 3; i++) {  
    max[i] = mark[0][i];  
    min[i] = mark[0][i];  
    for (j = 0; j < 8 ; j++) {  
  
        if (max[i] < mark[j][i]) {  
            max[i] = mark[j][i];  
        }  
  
        if (min[i] > mark[j][i]) {  
            min[i] = mark[j][i];  
        }  
    }  
}
```

Excel からのファイル入力(6)

```
for (i = 0; i < 3; i++) {  
    printf("%s : 最高点 %d 最低点 %d\n",  
        sub[i], max[i], min[i]);  
}  
  
fclose(fp);  
  
return 0;  
  
}
```


実行結果



```
コマンドプロンプト
Microsoft Windows [Version 10.0.19042.1055]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Nakano>cd C:\Users\Nakano\source\repos\file_test4\Debug

C:\Users\Nakano\source\repos\file_test4\Debug>file_test4
国語 : 最高点 76 最低点 76
数学 : 最高点 92 最低点 92
英語 : 最高点 84 最低点 84

C:\Users\Nakano\source\repos\file_test4\Debug>
```