



ce-5. 中間まとめ 1

(C プログラミング応用) (全 1 4 回)

URL: <https://www.kkaneko.jp/pro/c/index.html>

金子邦彦





- 変数の種類
 - 前回授業のプログラムを使って確認
 - int 整数
 - char 文字
 - double 浮動小数 など
- 論理的エラーの発見と解決
 - ステップ実行機能なども使用して，論理的エラーの発見と解決の練習を行う
- 条件分岐と繰り返しに関する練習
 - while と if の組み合わせ



変数の種類

```
#include "stdio.h"
#include <math.h>
#pragma warning(disable:4996)
int main()
{
    double x;
    double y;
    char buf[256];
    int i;
    double start_x;
    double step_x;
    FILE* fp;
    printf( "start_x =" );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%lf%n", &start_x );
    printf( "step_x =" );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%lf%n", &step_x );
    fp = fopen( "d:¥¥data.csv", "w" );
    for( i = 0; i < 20; i++ ) {
        x = start_x + ( i * step_x );
        y = sin( x );
        printf( "x= %f, y= %f%n", x, y );
        fprintf( fp, "x=, %f, y=, %f%n", x, y );
    }
    fprintf( stderr, "file d:¥¥data.csv created¥n" );
    fclose( fp );
    return 0;
}
```

変数は4種類使っている



整数を扱う int 型

整数は, 5, -3, 0 など

文字を扱う char 型

文字は, 1, 0, 3, -, a など

数字(1, 0, 3 など)も文字の一種

浮動小数を扱う double 型

3.14, -1.414, 5, 0, -3 など

(5, 0, -3 などの整数も浮動小数の一種)

ファイルポインタ

ファイル操作に使う変数



論理的エラーの発見と解決

例題 1 . 論理的エラーの例 (1)



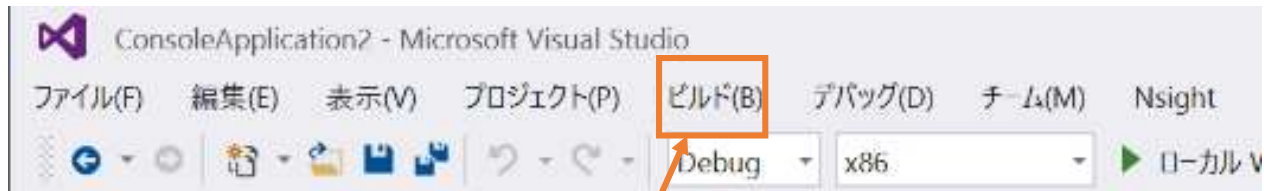
- 次ページのプログラムは，構文エラーは無い（ビルドは問題無くできる）
- 実行してみると，動作がおかしい
- 資料の手順に従って，原因を探し，解決しなさい



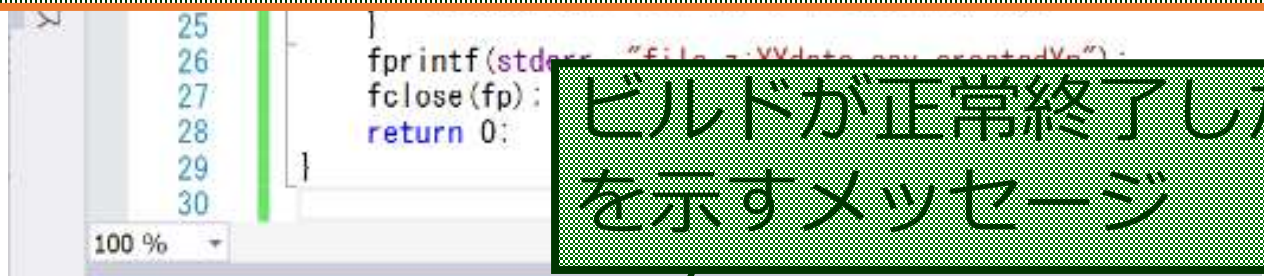
```
#include "stdio.h"
#include <math.h>
#pragma warning(disable:4996)
int main()
{
    double x;
    double y;
    char buf[256];
    int i;
    double start_x;
    double step_x;
    FILE* fp;
    printf( "start_x =" );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%lf%n", &start_x );
    printf( "step_x =" );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%lf%n", &step_x );
    fp = fopen( "d:¥¥data.csv", "w" );
    for( i = 0; i < 20; i++ ) {
        x = start_x + ( i * step_x );
        y = sin( x );
        printf( "x= %d, y= %d%n", x, y );
        fprintf( fp, "x=, %d, y=, %d%n", x, y );
    }
    fprintf( stderr, "file d:¥¥data.csv created¥n" );
    fclose( fp );
    return 0;
}
```

まずは、このページのプログラムを
ビルド、実行してみなさい

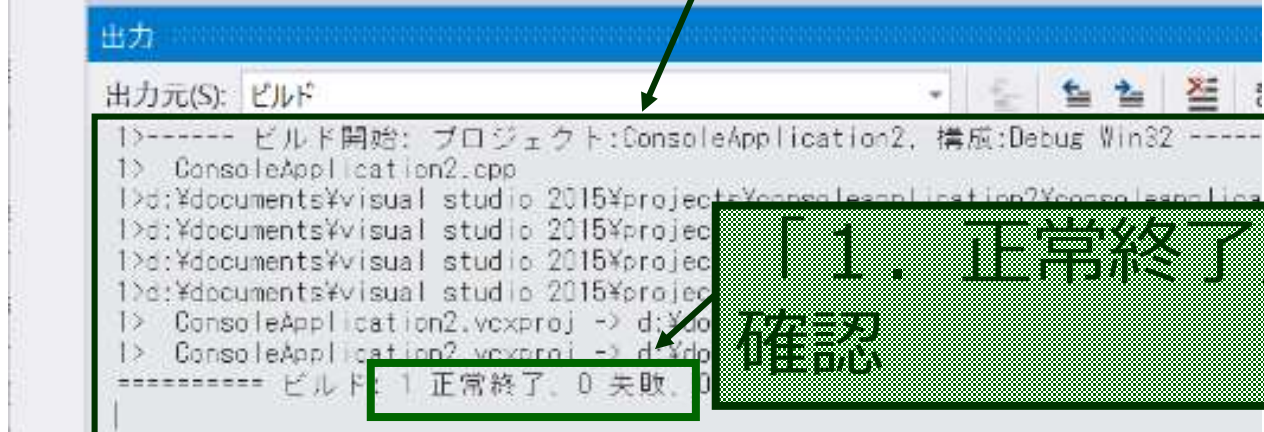
ビルド後の画面



ビルドの手順：
「ビルド」→「ソリューションのビルド」



ビルドが正常終了したことを示すメッセージ

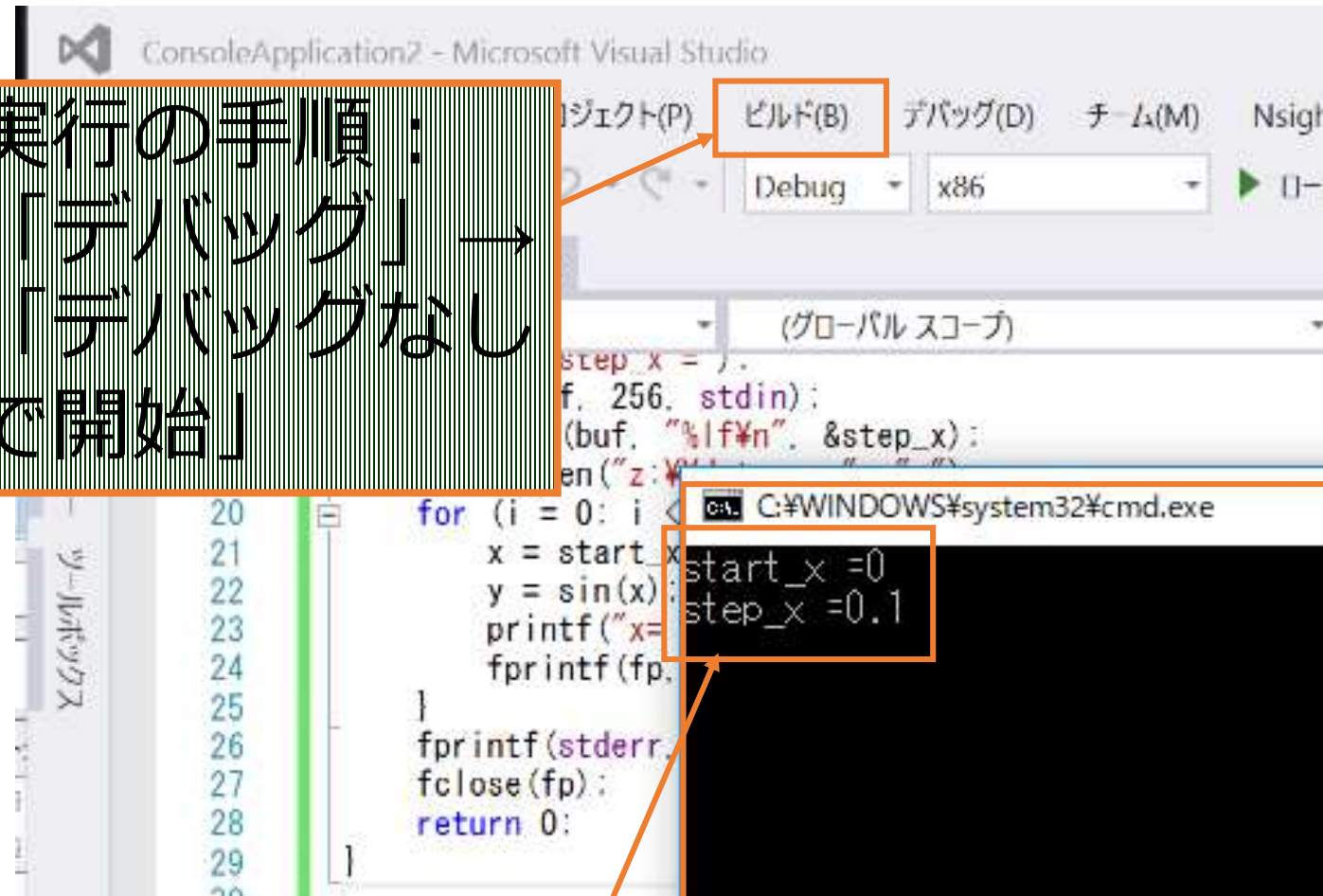


「1. 正常終了」を確認

実行中の画面



実行の手順：
「デバッグ」→
「デバッグなし
で開始」

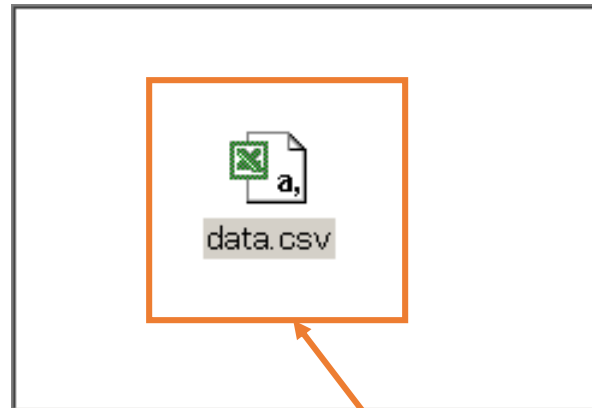


実行ウィンドウが現れるので、
start_x = 0 Enter
step_x = 0.1 Enter
のように操作してみる

実行結果の確認



実行の結果, データファイルが生成されるので,
中身を確認してみる



dドライブの
data.csv をダブル
クリックする



Microsoft Excel - data.csv

ファイル(F) 編集(E) 表示(V) 挿入(I) 書式(O) ツール(T) データ(D) ウィンドウ(W) ヘルプ(H) Adobe PDF(B)

MS Pゴシック 11 B I U

校閲結果の返信(C)... 校閲結果の差し込み終了(N)

	A	B	C	D	E	F	G	H	I
4	x=	8.59E+08	y=	1.07E+09					
5	x=	-1.7E+09	y=	1.07E+09					
6	x=	0	y=	1.07E+09					
7	x=	8.59E+08	y=	1.07E+09					
8	x=	1.72E+09	y=	1.07E+09					
9	x=	-1.7E+09	y=	1.07E+09					
10	x=	-8.6E+08	y=	1.07E+09					
11	x=	0	y=	1.07E+09					
12	x=	-1.7E+09	y=	1.07E+09					
13	x=	8.59E+08	y=	1.07E+09					
14	x=	-8.6E+08	y=	1.07E+09					
15	x=	1.72E+09	y=	1.07E+09					
16	x=	0	y=	1.07E+09					
17	x=	-1.7E+09	y=	1.07E+09					
18	x=	8.59E+08	y=	1.07E+09					
19	x=	-8.6E+08	y=	1.07E+09					
20	x=	1.72E+09	y=	1.07E+09					
21									
22									
23									
24									
25									

結果が期待通りになっていない

コマンド

```
#include "stdio.h"
#include <math.h>
#pragma warning(disable:4996)
int main()
{
    double x;
    double y;
    char buf[256];
    int i;
    double start_x;
    double step_x;
    FILE* fp;
    printf( "start_x =" );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%lf%n", &start_x );
    printf( "step_x =" );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%lf%n", &step_x );
    fp = fopen( "d:¥¥data.csv", "w" );
    for( i = 0; i < 20; i++ ) {
        x = start_x + ( i * step_x );
        y = sin( x );
        printf( "x= %f, y= %f\n", x, y );
        fprintf( fp, "x=, %f, y=, %f\n", x, y );
    }
    fprintf( stderr, "file d:¥¥data.csv created\n" );
    fclose( fp );
    return 0;
}
```

元のプログラムに間違いがあることが分かった。



x, y は **double** 型の変数なので、**sscanf** では「%lf」を、**printf**, **fprintf** では「%f」を使う決まりになっている

「%d」ではなく、「%f」が正しい



例題 2. 繰り返しと条件分岐

- キーボードから数値（正の整数）を読み込んで、足し算を続けるプログラム
- 変数の用途
 - sum 足し算の結果
 - buf キーボードから読み込んだ 1 行分
 - n キーボードから読み込んだ整数値



```
#include "stdio.h"
#include <math.h>
int main()
{
    int sum;
    int n;
    char buf[256];
    printf( "整数の足し算を続けます\n" );
    printf( "終了したいときは, 負の数を入力してください\n" );
    sum = 0;
    while ( 1 ) {
        printf( "整数値をどうぞ : ", buf );
        fgets( buf, 256, stdin );
        sscanf_s( buf, "%d\n", &n );
        if ( n >= 0 ) {
            printf ( "%d + %d = %d\n", sum, n, sum + n );
            sum = sum + n;
        }
        else {
            break;
        }
    }
    return 0;
}
```



kougil - Microsoft Visual C++ [デザイン] - kougil.cpp

ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)

Debug

リソース ビ... 4 x

スタート ページ kougil.cpp | print.c | _sftbuf.c | isatty.c | dbgheap.c | dbghook.c | malloc.c | crt0.c | < > x

グローバル tmain

```
#include "stdafx.h"
#include <math.h>
int _tmain(int argc, _TCHAR* argv[])
{
    int sum;
    int n;
    char buf[256];
    printf( "整数の足し算を続けます\n" );
    printf( "終了したいときは、負の数を入力してください\n" );
    sum = 0;
    while ( 1 ) {
        printf( "整数値をどうぞ：", buf );
        fgets( buf, 256, stdin );
        sscanf( buf, "%d\n", &n );
        if ( n >= 0 ) {
            printf( "%d + %d = %d\n", sum, n, sum + n );
            sum = sum + n;
        }
        else {
            break;
        }
    }
    return 0;
}
```

出力

ビルド

----- 終了 -----

ビルド : 1 正常終了、0 失敗、0 スキップ

コマンド ウィンドウ 出力 検索結果

ブレークポイント

コマンド

実行画面



```
c:\Documents and Settings\kaneko\My Documents\Visual Studio Projects\kougil\Debug\kougil.exe
整数の足し算を続けます
終了したいときは、負の数を入力してください
整数値をどうぞ：3
0 + 3 = 3
整数値をどうぞ：5
3 + 5 = 8
整数値をどうぞ：2
8 + 2 = 10
整数値をどうぞ：8
10 + 8 = 18
整数値をどうぞ：120
18 + 120 = 138
整数値をどうぞ：4
138 + 4 = 142
整数値をどうぞ：3
142 + 3 = 145
整数値をどうぞ：■
```


実行画面



```
c:\Documents and Settings\kaneko\My Documents\Visual Studio Projects\kougil\Debug\kougil.exe
整数の足し算を続けます
終了したいときは、負の数を入力してください
整数値をどうぞ：3
0 + 3 = 3
整数値をどうぞ：5
3 + 5 = 8
整数値をどうぞ：2
8 + 2 = 10
整数値をどうぞ：8
10 + 8 = 18
整数値をどうぞ：120
18 + 120 = 138
整数値をどうぞ：4
138 + 4 = 142
整数値をどうぞ：3
142 + 3 = 145
整数値をどうぞ：-2
```

「-2 Enter」のようにして、負の数を与えると、プログラムの実行が止まる

```
#include "stdio.h"
#include <math.h>
int main()
{
    int sum;
    int n;
    char buf[256];
    printf( "整数の足し算を続けます¥n" );
    printf( "終了したいときは, 負の数を入力してください¥n" );
    sum = 0;
    while ( 1 ) {
        printf( "整数値をどうぞ : ", buf );
        fgets( buf, 256, stdin );
        sscanf_s( buf, "%d¥n", &n );
        if ( n >= 0 ) {
            printf ( "%d + %d = %d¥n", sum, n, sum + n );
            sum = sum + n;
        }
        else {
            break;
        }
    }
    return 0;
}
```

「**while (1)**」は, 無条件に繰り返す
という意味になる
(「1」が, 常に成り立つ条件式の意味)

$n \geq 0$ のとき実行される部分

$n < 0$ のとき実行される部分

この「**break;**」は, **while** による繰り返し処理
から抜け出すという意味になる。
($n < 0$ のとき抜け出す)



- プログラムの作成と動作確認を行いなさい。次の 1, 2, 3 のうち 1 つ以上を各自で選ぶこと。
- 例題 2 のプログラムについて, 計算結果を, データファイル 「d:¥sum.csv」 に出力するように書き換えなさい
- for あるいは while を使い, $\sum_{k=1}^n k(k+1)$ を求めるプログラムを作成しなさい
- 例題 3 のプログラムについて, 平均値を表示できるように書き換えなさい