

ce-14. プログラムの実行順序

(Cプログラミング応用) (全14回)

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

金子邦彦



自由落下距離



- 前回の授業の「例題 1」の復習と重要事項の確認
 - 地上で物を落とし始めた後の自由落下距離を求める
 - 重力加速度 g は 9.8 とする
 - 自由落下距離を求めるために、プログラム中に、計算式 $y = (9.8 / 2.0) * x * x$ を書く

Microsoft Visual Studio C++ の画面構成



The screenshot displays the Microsoft Visual Studio C++ IDE interface. The main window shows the source code editor with the following code:

```
1 // ConsoleApplication1.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
2 //
3
4 #include "stdafx.h"
5
6
7 int main()
8 {
9
10
11
12 }
```

Three callout boxes provide additional information:

- C++ソースファイルの編集はここで行う** (C++ source file editing is done here)
- ファイルなどが表示される** (Files, etc., are displayed)
- ビルド結果が現れる** (Build results appear)

The Solution Explorer on the right shows the project structure:

- ソリューション 'ConsoleApplication1' (1 プロジェクト)
- ConsoleApplication1
 - 参照
 - 外部依存関係
 - ソース ファイル
 - ConsoleApplication1.cpp
 - stdafx.cpp
 - ヘッダー ファイル
 - stdafx.h
 - targetverh
 - リソース ファイル
 - ReadMe.txt

The Output window at the bottom left is empty, indicating that the build results have not yet been displayed.

```
#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 = ( 9.8 / 2.0 ) * x * 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;
}
```

**データファイル名
d:¥¥data.csv
は適切に設定すること**

自由落下距離の
計算を行っている部分

```
#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:%nndata.csv", "w" );
    for( i = 0; i < 20; i++ ) {
        x = start_x + ( i * step_x );
        y = ( 9.8 / 2.0 ) * x * x;
        printf( "x= %f, y= %f\n", x, y );
        fprintf( fp, "x=, %f, y=, %f\n", x, y );
    }
    fprintf( stderr, "file d:%nndata.csv created\n" );
    fclose( fp );
    return 0;
}
```

キーボードからの
データ読み込みを
行っている部分

計算を行っている部分

ファイルへの書き出し
を行っている部分

```
#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 = ( 9.8 / 2.0 ) * x * 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;
}
```

Cプログラムはメイン関数から
実行開始



変数 **x, y, buf, i, start_x,**
step_x, fp をメモリエリア中に確保

プログラムは順次実行

printf でメッセージを表示
fgets でキーボードから1行を読み込み
sscanf で数値を読み取って変数に格納

printf でメッセージを表示
fgets でキーボードから1行を読み込み
sscanf で数値を読み取って変数に格納

20回の繰り返し ($i = 0, 1, \dots, 19$)

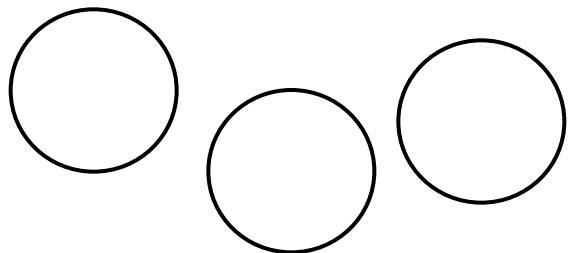
x の値から
 $(9.8 / 2.0) * x * x$
を求め、**y** に書き込む

「ビルド」による実行ファイルの生成



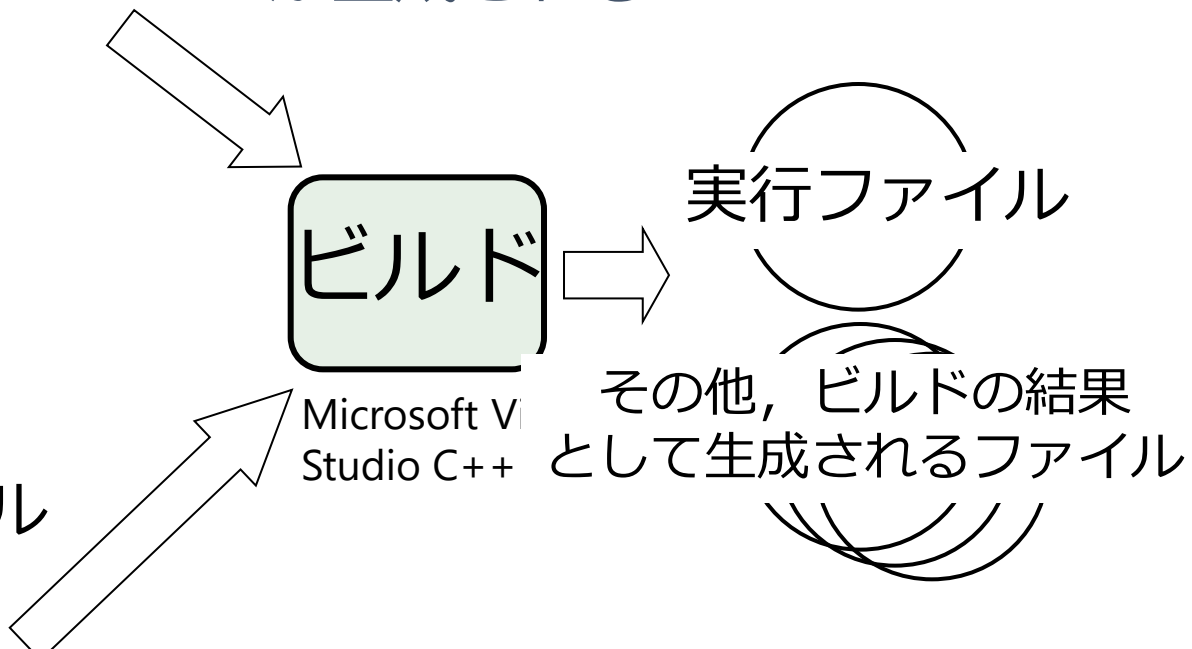
```
#include "stdio.h"
#include <math.h>
#pragma warning(disable:4996)
int main()
{
    double x;
    double y;
    char buf[256];
    int i;
    double start_x;
    double step_x;
    FILE* fp;
    printf( "start_x = " );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%lf\n", &start_x );
    printf( "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 = ( 9.8 / 2.0 ) * x * 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;
}
```

C++ソースファイル



その他のファイル

「ビルド」により実行ファイル等
が生成される



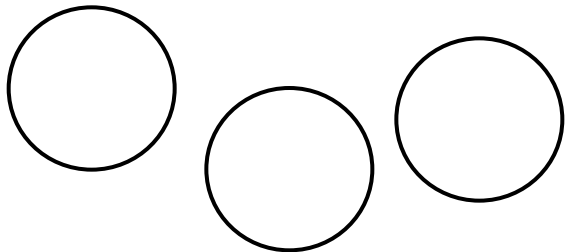
構文エラーがあると,
「ビルド」時にエラーメッセージが表示される
(実行ファイルは生成されない)

「消去 (clean)」 の操作



```
#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 = ( 9.8 / 2.0 ) * x * 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;
}
```

C++ソースファイル



その他のファイル

ビルド

Microsoft Visual
Studio C++

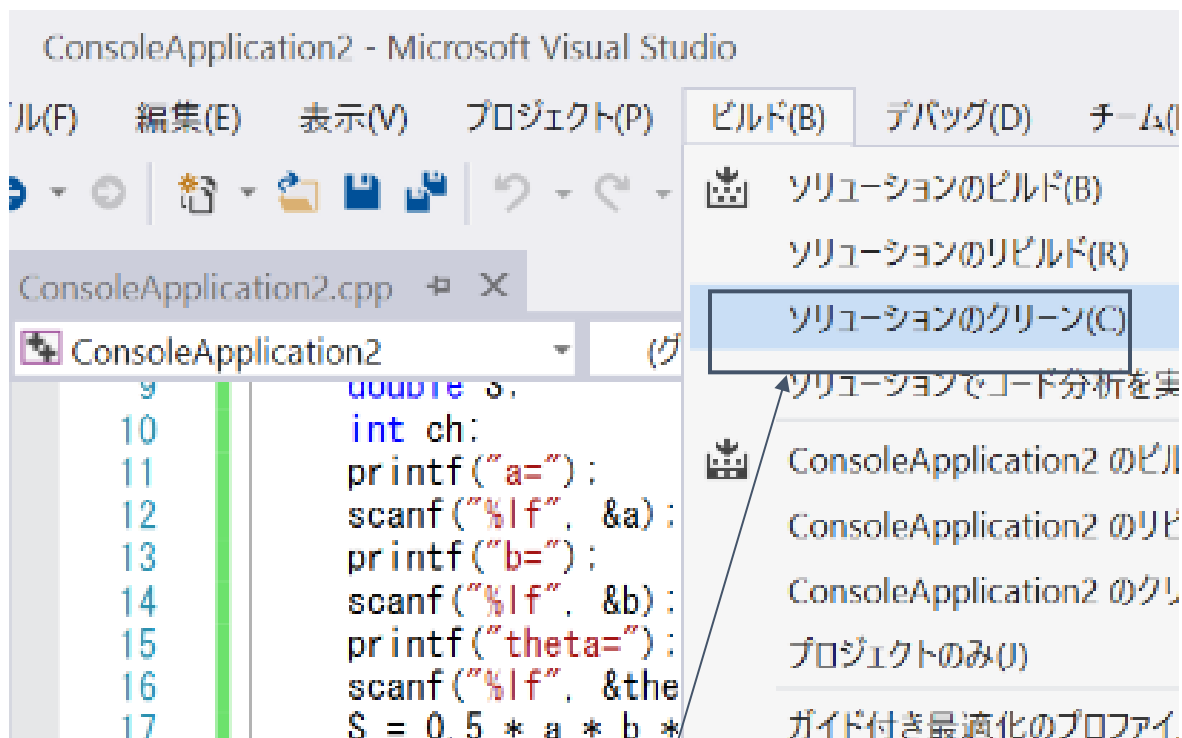
実行ファイル

その他, ビルドの結果
として生成されるファイル

「消去 (clean)」 の操作により,
ビルドの結果が消える

(元のソースファイル等は消えない) • 8

「消去 (clean) 」の操作



「消去 (clean) 」の操作を行っても、元のソースファイルが消えるわけではない (ディスク使用量の節約になる)

構文エラーの例



- 全角文字
 - 全角文字はプログラム中の決められた場所にしか入れてはならない。
 - コメント
 - 文字列（ダブルクォートで囲まれた部分）
- 「"」 抜け
- 「;」 忘れ
- など

ビルドが正常終了し、実行ファイルが生成できたことを示すメッセージ

```
16 scanf("%lf", &theta);  
17 S = 0.5 * a * b * sin(theta * 3.14159 / 180.0);  
18 printf("S = %f¥n", S);  
19 ch = getchar();  
20 ch = getchar();  
21 return 0;  
22 }  
23
```

ビルド結果が現れる。「1 正常終了, 0 失敗・・・」ならばビルドに成功

出力
出力元(S): ビルド

```
1>----- すべてのリビルド開始: プロジェクト:ConsoleApplication2, 構成:Debug Win32 -----  
1> stdafx.cpp  
1> ConsoleApplication2.cpp  
1> ConsoleApplication2.vcxproj -> d:¥documents¥visual studio 2015¥Projects¥ConsoleApplica  
1> ConsoleApplication2.vcxproj -> d:¥documents¥visual studio 2015¥Projects¥ConsoleApplica  
----- すべてのリビルド 1 正常終了, 0 失敗, 0 スキップ -----
```

構文エラーの例 (1)



```
ConsoleApplication2 - Microsoft Visual Studio
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) チーム(M) Nsight ツール(T) アーキテ
Debug x86 ローカル Windows デバッガー
ConsoleApplication2.cpp
ConsoleApplication2
1 #include <math.h>
2 #include <string.h>
3 #pragma warning(disable:4996)
4 int _tmain
5 {
6     double a;
7     double b;
8     double theta;
9     double S;
10    int ch;
11    printf("a=");
```

ここでは、セミコロン「;」を入れるのを忘れていた (人間の目では発見が難しい)

ビルドが失敗したことを示すメッセージ

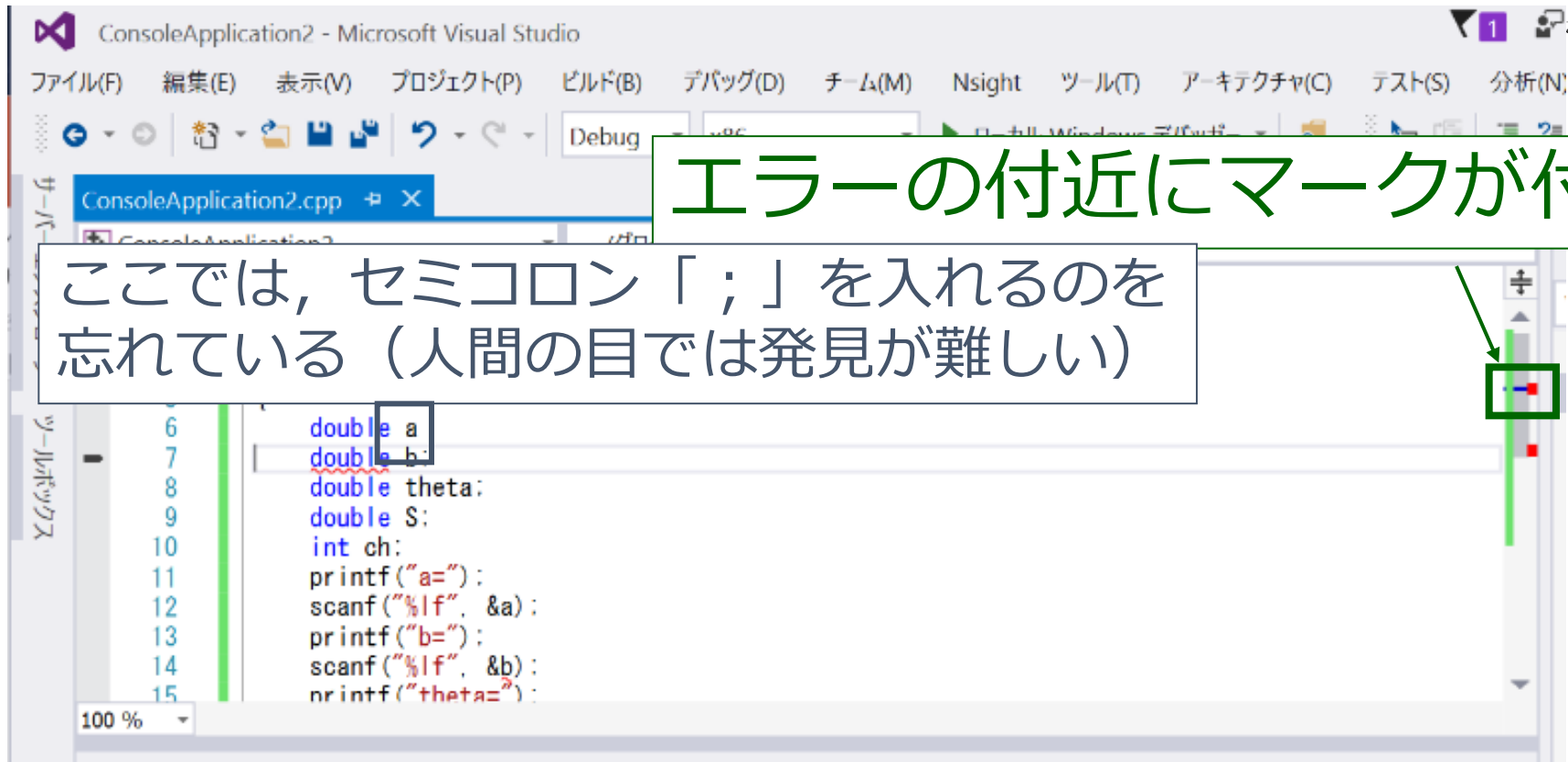
100 %

ソリューション全体 3 エラー 0 警告 0 メッセージ

コード	説明	プロジェクト	ファイル	行
abc	識別子 "a" は必要です	ConsoleApplicatio	ConsoleApplicatio	7
abc	識別子 "b" が定義されていません	ConsoleApplicatio	ConsoleApplicatio	14
C2144	構文エラー: 'double' は ';' によって先行されなければなりません。	ConsoleApplicatio	consoleapplicatio	7

7行目に構文エラー

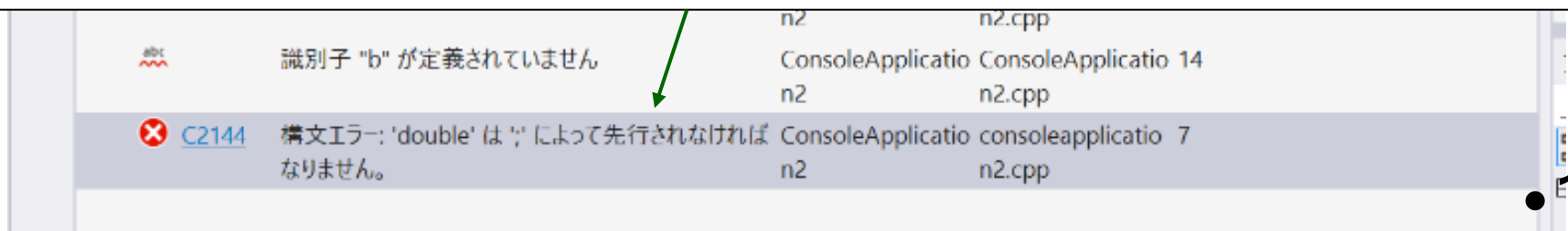
構文エラーの例 (1)



エラーの付近にマークが付く

ここでは、セミコロン「;」を入れるのを忘れている (人間の目では発見が難しい)

マウスで「構文エラー」とある行をダブルクリックすると、エディタのカーソルが動く



構文エラーの例 (2)



```
ConsoleApplication2 - Microsoft Visual Studio
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) チーム(M) Nsight ツール(T) アーキテクチャ(C) テスト(S)
Debug x86 ローカル Windows デバッガー
ConsoleApplication2.cpp
ConsoleApplication2 (グローバルスコープ) _tmain()
1 #include "stdafx.h"
2 #include <math.h>
3 #pragma warning(disable:4996)
4 int _tmain()
5 {
6     double a;
7     double b;
8     double theta;
9     doublr S;
10    int on;
11    printf("a=");
12    scanf("%lf", &a);
13    printf("b=");
```

ここで、「doublr」とあるのはスペルミス
(正しくは、double)

ビルドが失敗したことを示すメッセージ
(ミスは1箇所なのに、エラーは多数)

コード	説明	プロジェクト	ファイル	行	抑制状態
	識別子 "doublr" が定義されていません	ConsoleApplicatio	ConsoleApplicatio	9	
✖ C2065	'doublr': 定義されていない識別子です。	ConsoleApplicatio	consoleapplicatio	9	
✖ C2146	構文エラー: ';' が、識別子 'S' の前に必要です。	ConsoleApplicatio	consoleapplicatio	9	
✖ C2065	'S': 定義されていない識別子です。	ConsoleApplicatio	consoleapplicatio	9	
✖ C2065	'S': 定義されてい				
✖ C2065	'S': 定義されていない識別子です。	ConsoleApplicatio	consoleapplicatio	18	

9行目に「'doublr': 定義されていない識別子です」

データファイル



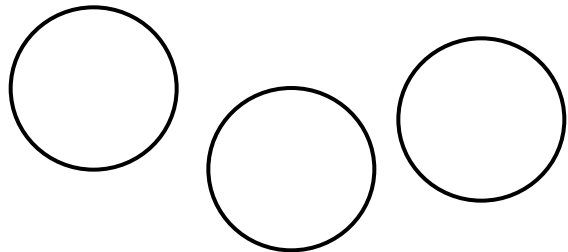
- プログラムでのファイル操作
 - ファイル生成
 - ファイル読み出し
 - ファイル書き込み

「自由落下距離」のプログラムを実行すると データファイルが生成される

```
#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);  
    p = fopen("d:\%data.csv", "w");  
    for(i = 0, i < 20, i++) {  
        x = start_x + (i * step_x);  
        y = (9.8 / 2.0) * x * 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;  
}
```

データファイルに関係
している部分

C++ソースファイル



その他のファイル

ビルド

実行ファイル

データファイル

自由落下距離のプログラムを実行すると
データファイル data.csv が生成される

例題 1 .自由落下距離



- 「自由落下距離」のプログラムについて，実行順を確認するとともに，変数の値の変化を観察する
 - Microsoft Visual Studio C++ のステップ実行機能を使用
 - 前回の授業で作成した「プロジェクト」を開く

```
#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 = ( 9.8 / 2.0 ) * x * 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;
}
```

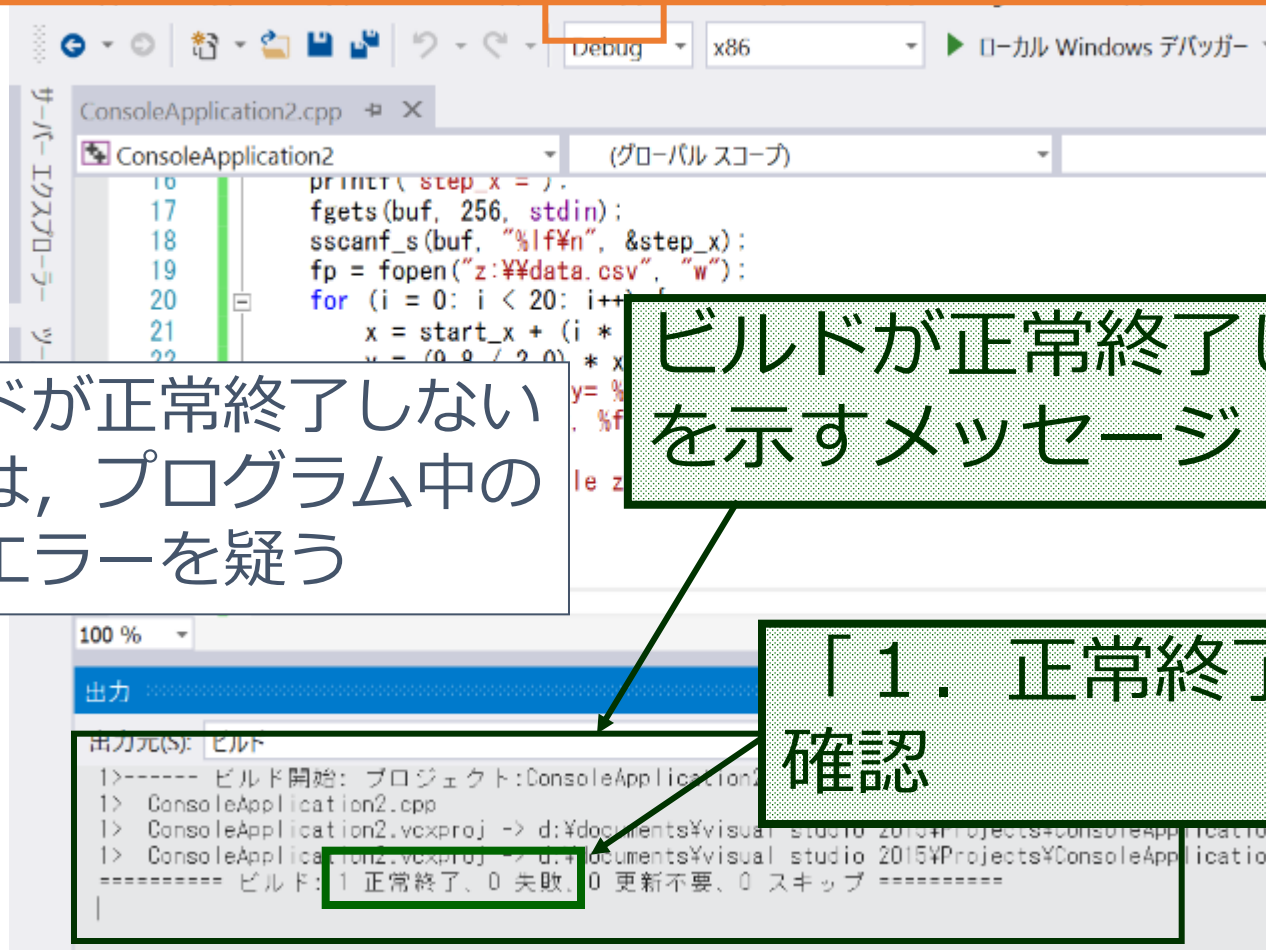
**データファイル名
d:¥¥data.csv
は適切に設定すること**

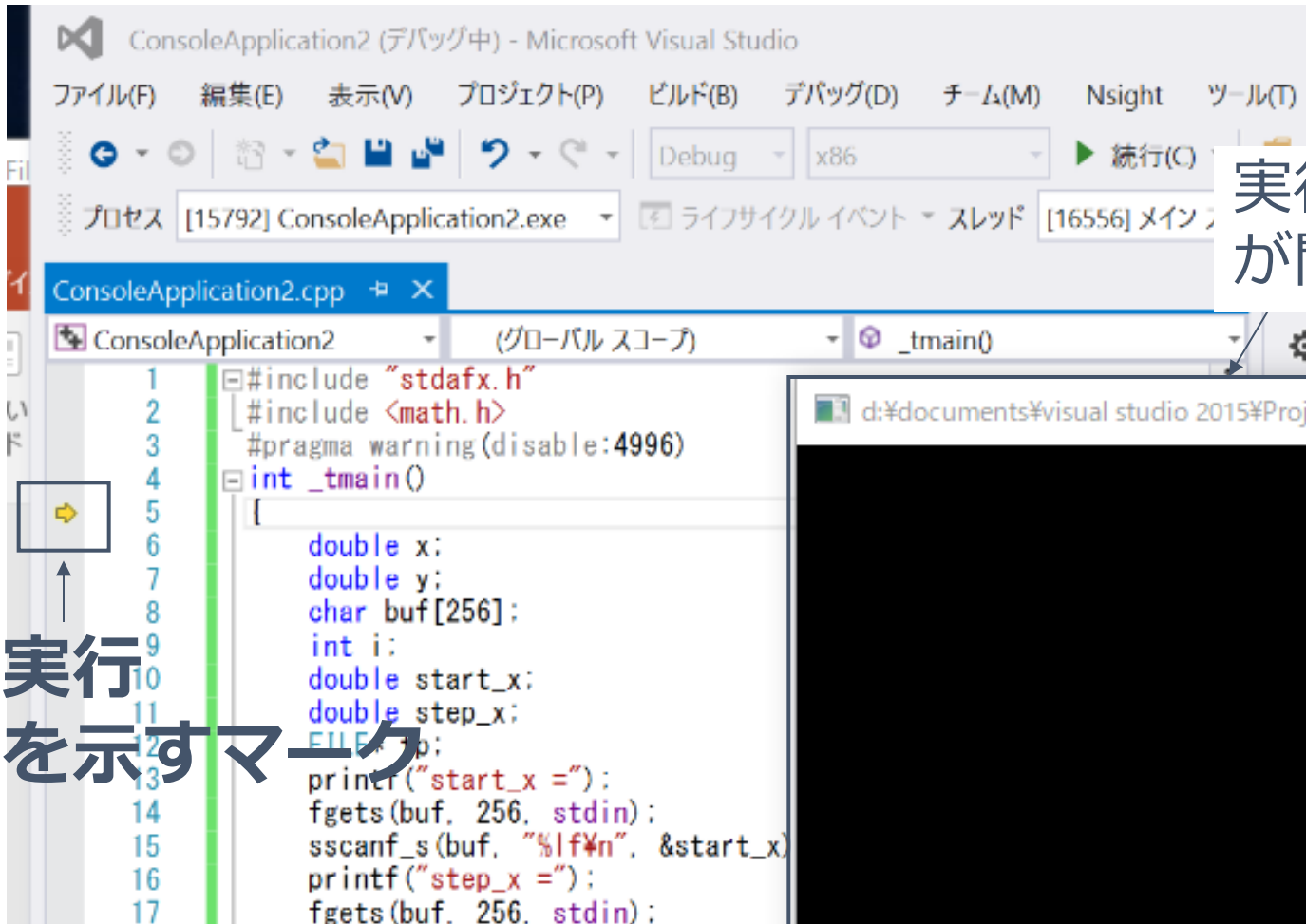
自由落下距離の
計算を行っている部分

ビルド後の画面



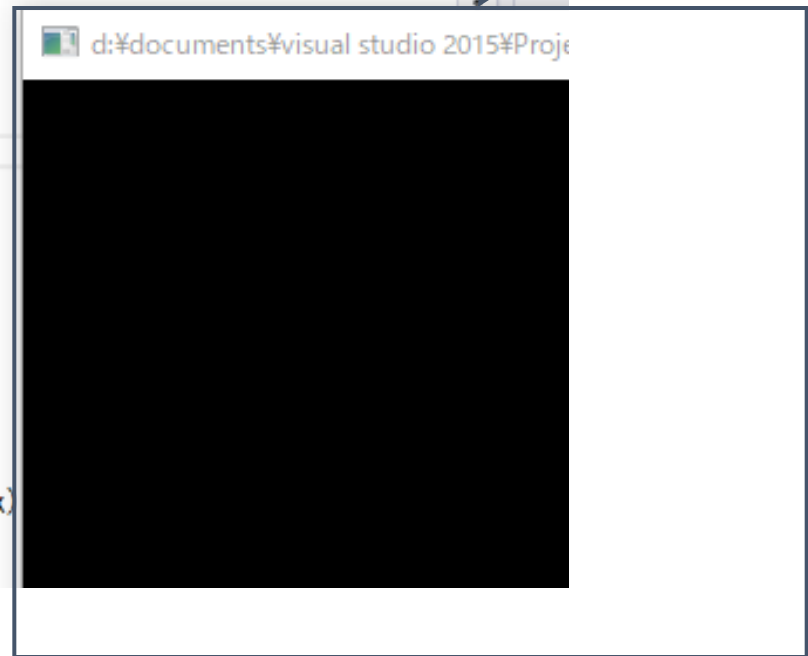
ビルドの手順：
「ビルド」 → 「ソリューションのビルド」



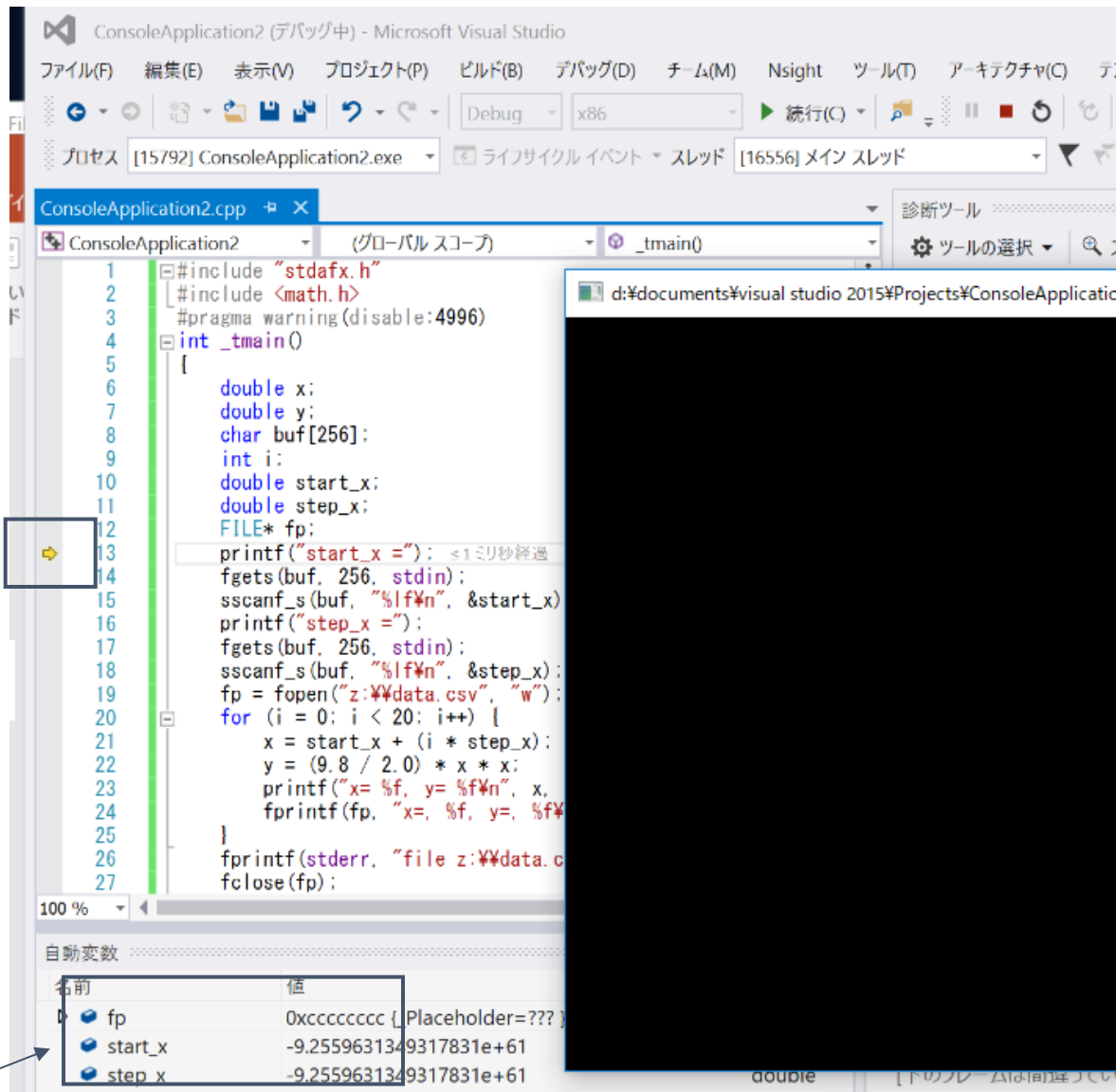


実行ウィンドウ
が開く

実行
を示すマーク



「F10」キーを押すとステップ実行が始まる。(マウスカーソルは、Microsoft Visual Studio C++ 内に入れておく)



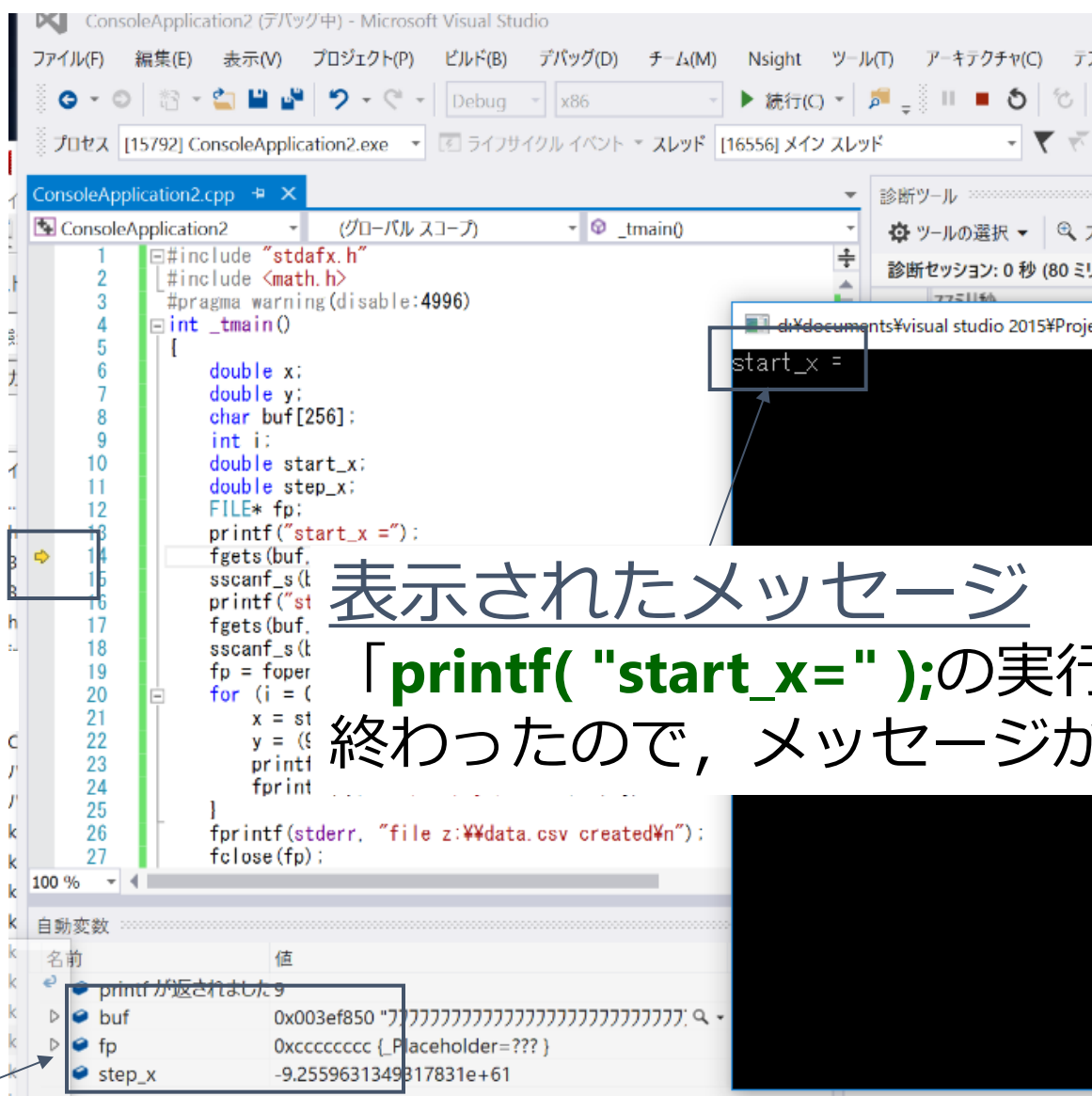
```
1 #include "stdafx.h"
2 #include <math.h>
3 #pragma warning(disable:4996)
4 int _tmain()
5 {
6     double x;
7     double y;
8     char buf[256];
9     int i;
10    double start_x;
11    double step_x;
12    FILE* fp;
13    printf("start_x ="): ≤1ミリ秒経過
14    fgets(buf, 256, stdin);
15    sscanf_s(buf, "%lf\n", &start_x)
16    printf("step_x =");
17    fgets(buf, 256, stdin);
18    sscanf_s(buf, "%lf\n", &step_x);
19    fp = fopen("z:¥¥data.csv", "w");
20    for (i = 0; i < 20; i++) {
21        x = start_x + (i * step_x);
22        y = (9.8 / 2.0) * x * x;
23        printf("x= %f, y= %f\n", x,
24            fprintf(fp, "x=, %f, y=, %f¥¥\n", x, y);
25    }
26    fprintf(stderr, "file z:¥¥data.c
27    fclose(fp);
```

名前	値
fp	0xcccccccc { Placeholder=??? }
start_x	-9.2559631349317831e+61
step_x	-9.2559631349317831e+61

マークが進む

変数の値を
観察できる

さらに「F10」キーを押すとス
テップ実行が続く



```
1 #include "stdafx.h"
2 #include <math.h>
3 #pragma warning(disable:4996)
4 int _tmain()
5 {
6     double x;
7     double y;
8     char buf[256];
9     int i;
10    double start_x;
11    double step_x;
12    FILE* fp;
13    printf("start_x =");
14    fgets(buf,
15         sscanf_s(t
16         printf("st
17         fgets(buf,
18         sscanf_s(t
19         fp = fopen
20         for (i = (
21             x = st
22             y = (
23             printf
24             fprint
25     }
26     fprintf(stderr, "file z:%%data.csv created\n");
27     fclose(fp);
```

診断ツール

ツールの選択

診断セッション: 0 秒 (80 ミリ)

自動変数

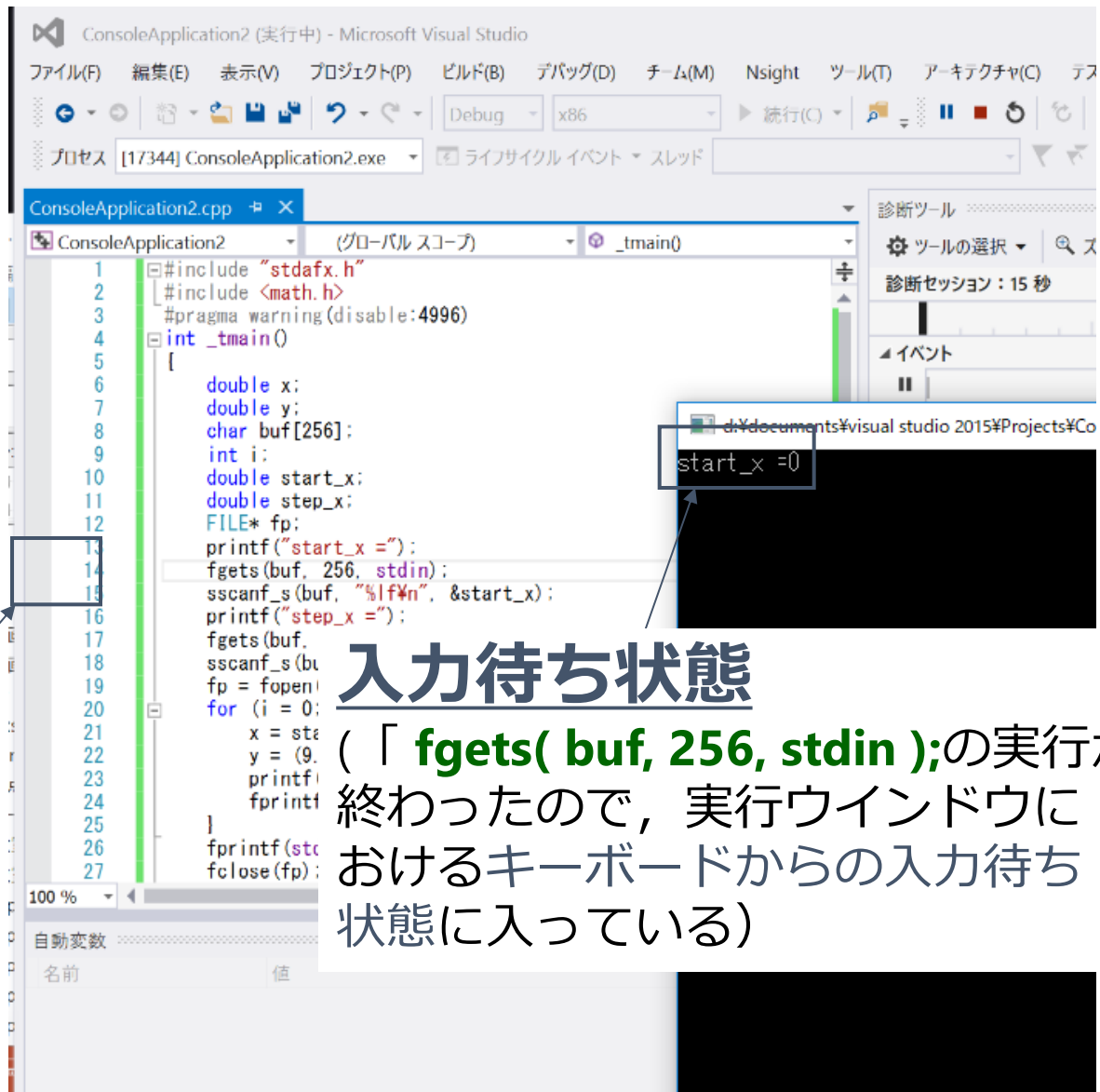
名前	値
printf が返されました	9
buf	0x003ef850 "???
fp	0xcccccccc [_Placeholder=???
step_x	-9.2559631349817831e+61

マークが進む

表示されたメッセージ
「**printf("start_x=");**の実行が
終わったので、メッセージが出る

変数の値を
観察できる

さらに「**F10**」キーを押すとス
テップ実行が続く

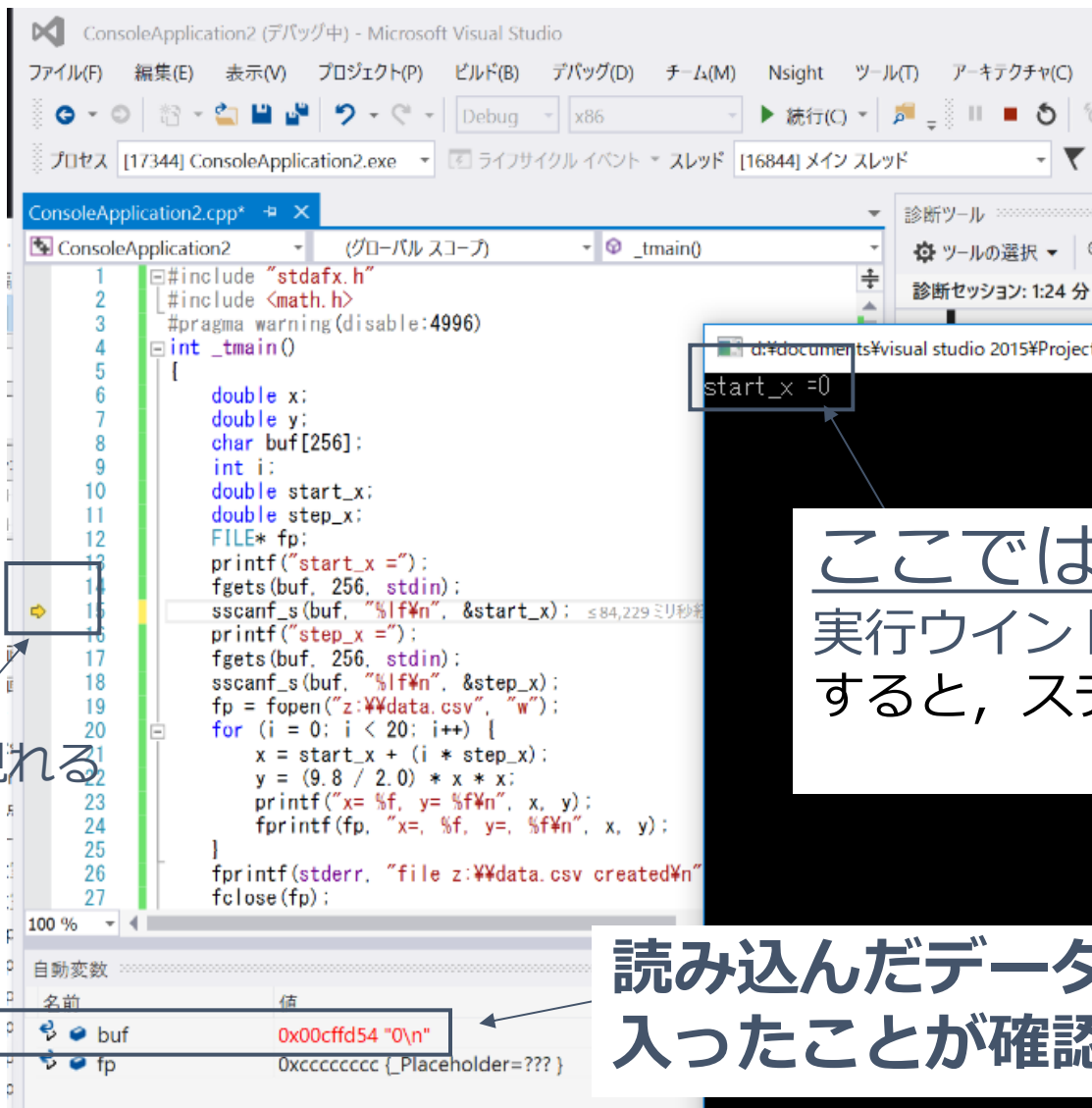


マークが
変化

入力待ち状態

(「**fgets(buf, 256, stdin);**」の実行が
終わったので、実行ウィンドウに
おけるキーボードからの入力待ち
状態に入っている)

さらに「**F10**」キーを押すとステップ実行
が続く (今度は入力待ち状態)



```
#include "stdafx.h"
#include <math.h>
#pragma warning(disable:4996)
int _tmain()
{
    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("z:\\data.csv", "w");
    for (i = 0; i < 20; i++) {
        x = start_x + (i * step_x);
        y = (9.8 / 2.0) * x * x;
        printf("x= %f, y= %f\\n", x, y);
        fprintf(fp, "x=, %f, y=, %f\\n", x, y);
    }
    fprintf(stderr, "file z:\\data.csv created\\n");
    fclose(fp);
}
```

自動変数

名前	値
buf	0x00cffd54 "0\\n"
fp	0xcccccccc {_Placeholder=??}

マークが現れる

ここでは 0 Enter
実行ウィンドウで、「0 Enter」と
すると、ステップ実行が再開する

読み込んだデータが変数 buf に
入ったことが確認できる

実行ウィンドウで「0 Enter」とすると実行
が続く (0 の部分は数値なら何でも良い)

実行手順 (通常実行の場合)



- Microsoft Visual Studio C++ で 「デバック」 → 「デバッグなしで開始」
 - すると, 新しいウィンドウが開く
- 新しいウィンドウが現れるので, `start_x`, `step_x` の値をキーボードから与える

- 例えば
 - `start_x = 0`
 - `step_x = 0.1`

A screenshot of a Windows command prompt window. The title bar reads "C:\Documents and Settings\kaneko\My Documents\Visual Studio Projects\koug1\Debug\koug1.exe". The window content shows the text "start_x = 0" and "step_x =" on separate lines, indicating that these values have been entered at the prompt.

- ウィンドウは消えるが, d: ドライブに `data.csv` (データファイル) が作成されるので, Excel 等で開き確認する

実行手順 (ステップ実行の場合)



- Microsoft Visual Studio C++ で
「F10(ファンクションの10)」
→ すると, 新しいウィンドウが開く

「F10」は, Microsoft Visual Studio C++のウィンドウ内
にマウスカーソルを入れた状態で押すこと
- 「F10」を押すたびに, 1ステップずつ実行が進む

例題 2. 平方根の計算



- 浮動小数データを読み込んで、平方根の計算と表示を行うプログラムを作る。
 - 但し、負の数の場合には、メッセージを表示する
 - 負の数であるかどうかによって条件分岐を行うために if 文を使う。

例) 9 のとき : 3

- 1 のとき : メッセージを表示

```
#include "stdio.h"
#include <math.h>
int main()
{
    double x;
    double y;
    char buf[256];
    int ch;
    printf("x=");
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%lf\n", &x );
    if ( x < 0 ) {
        printf("負なので計算できません\n");
    }
    else {
        y = sqrt(x);
        printf("sqrt(%f)=%f\n", x, y);
    }
    ch = getchar();
    ch = getchar();
    return 0;
}
```

条件式

条件が成り立つ場合に実行される部分

条件が成り立たない場合に実行される部分

実行順 ($x < 0$) の場合

```
#include "stdio.h"
#include <math.h>
int main()
{
    double x;
    double y;
    char buf[256];
    int ch;
    ① printf("x=");
    ② fgets( buf, 256, stdin );
    ③ sscanf_s( buf, "%lf%n", &x );
    if ( x < 0 ) {
    ④ printf("負なので計算できません\n");
    }
    else {
        y = sqrt(x);
        printf("sqrt(%f)=%f\n", x, y);
    }
    ⑤ ch = getchar();
    ⑥ ch = getchar();
    ⑦ return 0;
}
```

実行順 ($x \geq 0$) の場合

```
#include "stdio.h"
#include <math.h>
int main()
{
    double x;
    double y;
    char buf[256];
    int ch;
    ① printf("x=");
    ② fgets( buf, 256, stdin );
    ③ sscanf_s( buf, "%lf%n", &x );
    if ( x < 0 ) {
        printf("負なので計算できません\n");
    }
    else {
        ④ y = sqrt(x);
        ⑤ printf("sqrt(%f)=%f\n", x, y);
    }
    ⑥ ch = getchar();
    ⑦ ch = getchar();
    ⑧ return 0;
}
```

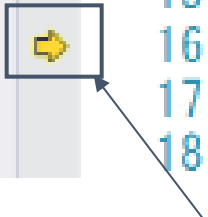
平方根の計算



```
ConsoleApplication2.cpp  X
ConsoleApplication2      (グローバル スコープ)      _tmain()
1  #include "stdafx.h"
2  #include <math.h>
3  int _tmain()
4  {
5      double x;
6      double y;
7      char buf[256];
8      int ch;
9      printf("x=");
10     fgets(buf, 256, stdin);
11     sscanf_s(buf, "%lf\n", &x);
12     if (x < 0) { s1ミリ秒経過
13         printf("負なので計算できません\n");
14     }
15     else {
```

マークが「if (x < 0) {」に来たときに、
「F10」を押すと

```
ConsoleApplication2.cpp  X
ConsoleApplication2 (グローバル スコープ) _tmain()
1 #include "stdafx.h"
2 #include <math.h>
3 int _tmain()
4 {
5     double x;
6     double y;
7     char buf[256];
8     int ch;
9     printf("x=");
10    fgets(buf, 256, stdin);
11    sscanf_s(buf, "%lf\n", &x);
12    if (x < 0) {
13        printf("負なので計算できません\n");
14    }
15    else {
16        y = sqrt(x); ≤1ミリ秒経過
17        printf("sqrt(%f)=%f\n", x, y);
18    }
}
```

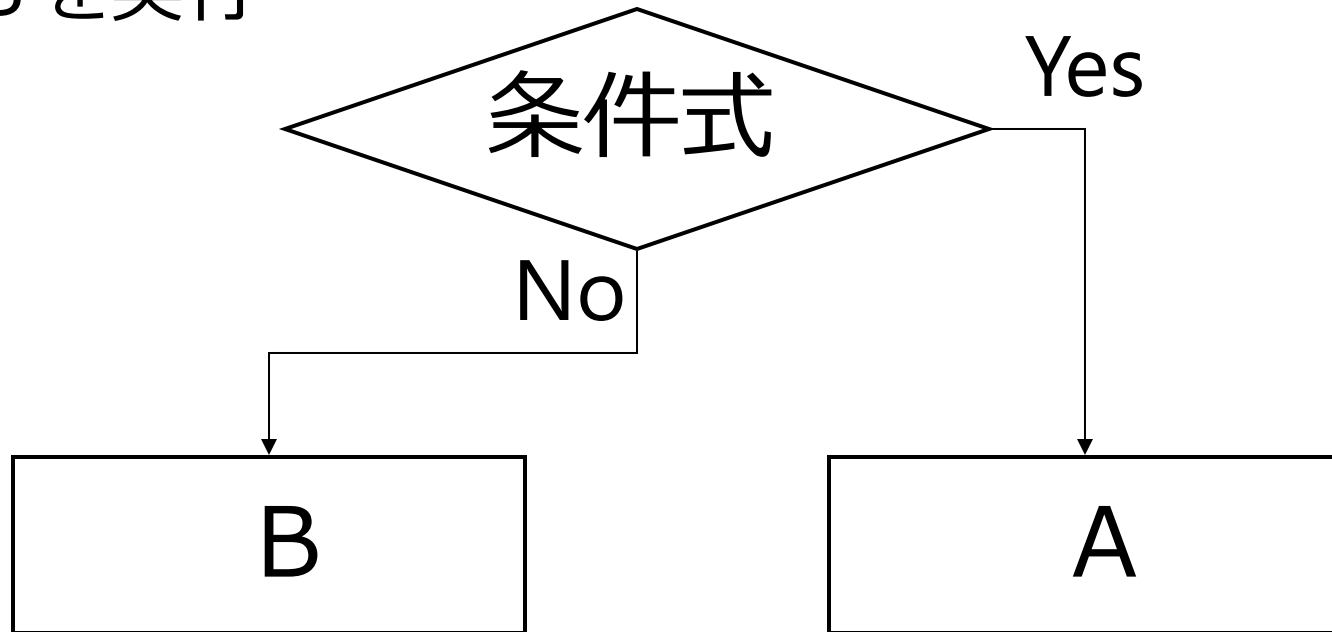


「y = sqrt(x);」 の行にジャンプする

条件分岐とは



- 「ある条件式」が成り立てばAを、成り立たなければBを実行

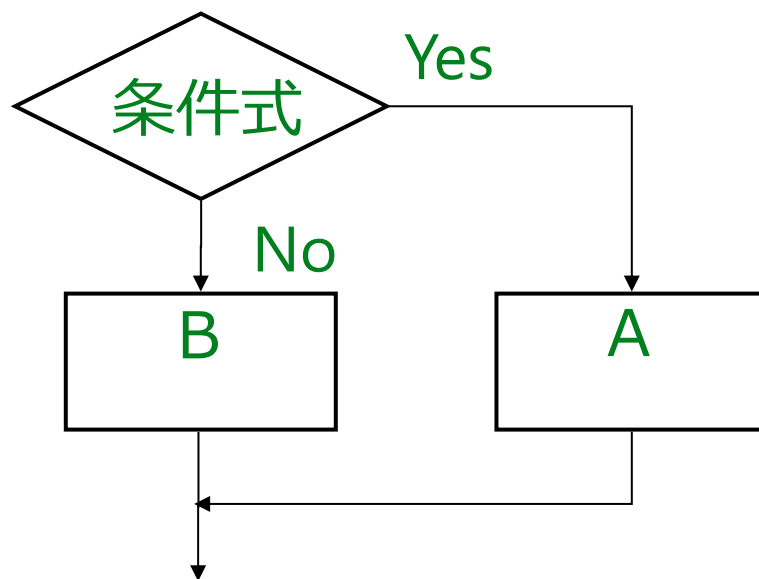


if 文と else 文



「条件式」が成り立てばAを、成り立たなければBを実行

```
if ( 条件式 ) {  
    式;  
    ... ;  
    A  
}  
else {  
    式;  
    ... ;  
    B  
}
```



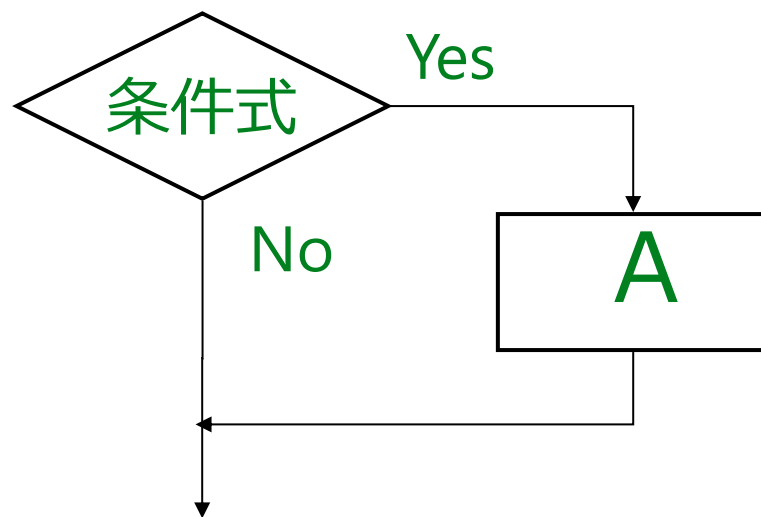
if 文



if 文のみを書いて, else 文を書かないこともできる.

「ある条件」が成り立つときに限り A を実行

```
if ( 条件式 ) {  
    式;  
    ... ;  
} A
```



- 条件式の中には、ふつう、比較演算を書く

- 演算子

意味

<

左辺が右辺より小さい

< =

左辺が右辺以下

>

左辺が右辺より大きい

> =

左辺が右辺以上

= =

左辺が右辺と等しい

! =

左辺が右辺と等しくない

比較演算の例



「左辺が右辺以上」の意味

```
if (age >= 20 ){  
    printf("You may drink alcoholic beverage. ");  
}  
else{  
    printf("You may not drink alcoholic  
beverage.");  
}
```

字下げとセミコロンを忘れないこと



- セミコロンを忘れると
 - プログラムは動かない

字下げ

```
if (条件式) {  
  文;  
  ...;  
}
```

セミコロン

- 字下げを忘れると
 - プログラムは動くが、読みづらい

字下げ

```
if (条件式) {  
  文;  
  ...;  
}  
else {  
  文;  
  ...;  
}
```

セミコロン

セミコロン

例題 3. 棒グラフでのステップ実行



- 整数から，その長さだけの棒を表示する関数 bar を作る
例) 5 → *****
- 関数 bar を使って，「整数を読み込んで，読み込んだ長さの棒を表示するメイン関数を作る
- ステップ実行（「F10」キー）を行う

棒グラフ



```
#include "stdio.h"
#include <math.h>
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
        printf("*");
    }
    printf("¥n");
    return;
}
int main()
{
    int len;
    char buf[256];
    int ch;
    printf( "len = " );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%d¥n", &len );
    bar( len );
    ch = getchar();
    ch = getchar();
    return 0;
}
```

bar関数

main関数

複数の関数を含む
プログラム

プログラム実行は
main 関数（メイン
関数）から始まる

プログラム実行順



```
#include "stdio.h"
#include <math.h>
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
        printf("*");
    }
    printf("¥n");
    return;
}
```

戻り

メイン関数の先頭行
がプログラム実行の始まり

```
int main()
{
    int len;
    char buf[256];
    int ch;
    ① printf( "len =" );
    ② fgets( buf, 256, stdin );
    ③ sscanf_s( buf, "%d¥n", &len );
    bar( len );
    ⑦ ch = getchar();
    ⑧ ch = getchar();
    ⑨ return 0;
}
```

関数呼び出し

メイン関数内の **return**
がプログラム実行の終わり

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」

```
1 #include "stdafx.h"
2 #include <math.h>
3 void bar(int len)
4 {
5     int i;
6     for (i = 0; i < len; i++)
```

ビルドが正常終了しないときは、プログラム中の構文エラーを疑う

ビルドが正常終了したことを示すメッセージ

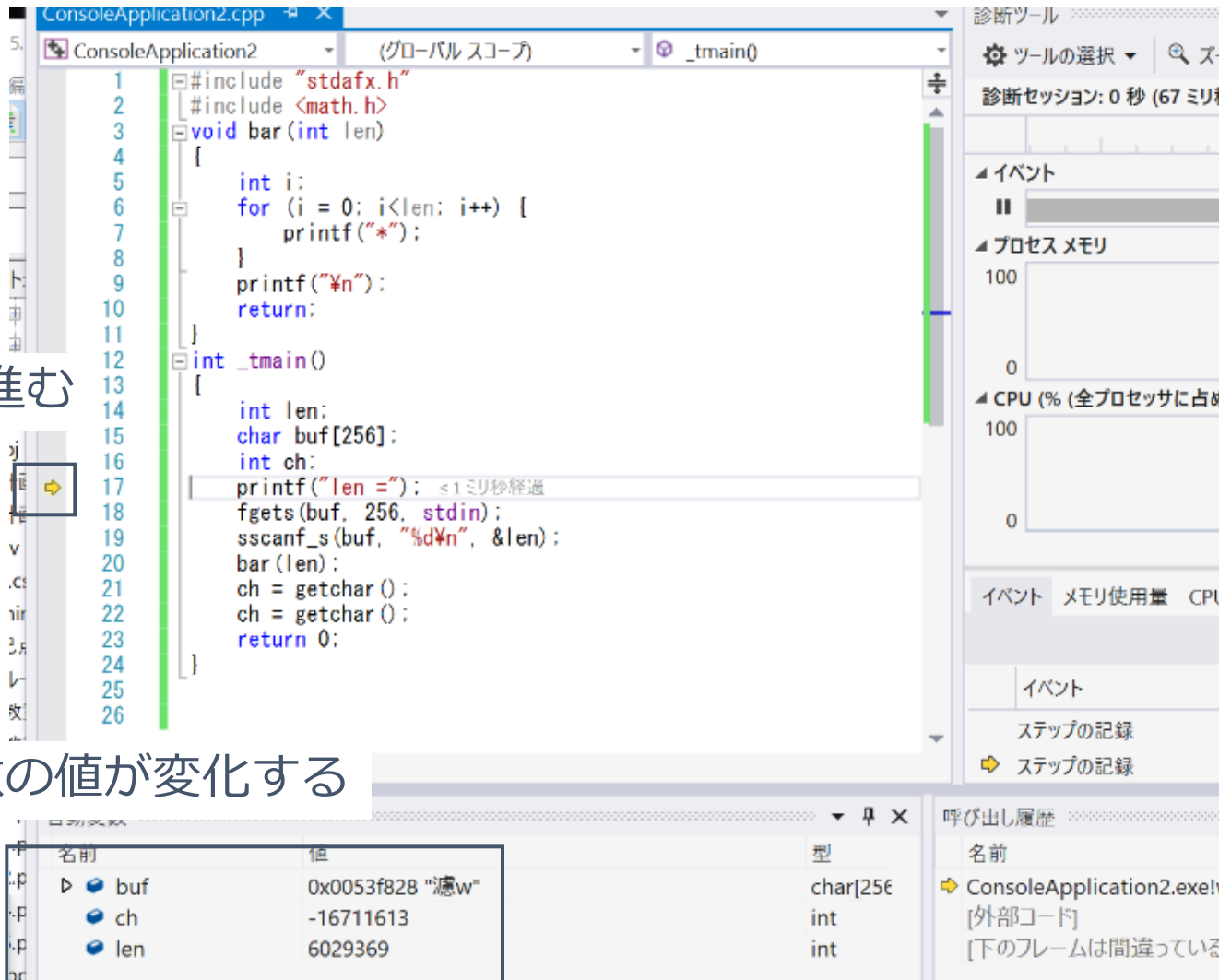
```
出力
出力元(S): ビルド
ビルド開始: プロジェクト: ConsoleApplication2, 構成: Debug Win82
1> ConsoleApplication2.cpp
1> ConsoleApplication2.vcxproj -> d:\documents\visual studio 2015\Projects\ConsoleApplication2\ConsoleApplication2.vcxproj
1> ConsoleApplication2.vcxproj -> d:\documents\visual studio 2015\Projects\ConsoleApplication2\ConsoleApplication2.vcxproj
***** ビルド: 1 正常終了, 0 失敗, 0 更新不要, 0 スキップ *****
```

ステップ実行



```
1 #include "stdafx.h"
2 #include <math.h>
3 void bar(int len)
4 {
5     int i;
6     for (i = 0; i < len; i++) {
7         printf("*");
8     }
9     printf("%n");
10    return;
11 }
12 int _tmain()
13 {
14     int len;
15     char buf[256];
16     int ch;
17     printf("len = ");
18     fgets(buf, 256, stdin);
19     sscanf_s(buf, "%d%n", &len);
20     bar(len);
21     ch = getchar();
22     ch = getchar();
23     return 0;
24 }
25
26
```

「F10」キーを押すとステップ実行が始まる。(マウスカーソルは、Microsoft Visual Studio C++ 内に入れておく)



ConsoleApplication2.cpp

```
1 #include "stdafx.h"
2 #include <math.h>
3 void bar(int len)
4 {
5     int i;
6     for (i = 0; i < len; i++) {
7         printf("*");
8     }
9     printf("\n");
10    return;
11 }
12 int _tmain()
13 {
14     int len;
15     char buf[256];
16     int ch;
17     printf("len = ");
18     fgets(buf, 256, stdin);
19     sscanf_s(buf, "%d\n", &len);
20     bar(len);
21     ch = getchar();
22     ch = getchar();
23     return 0;
24 }
25
26
```

マークが進む

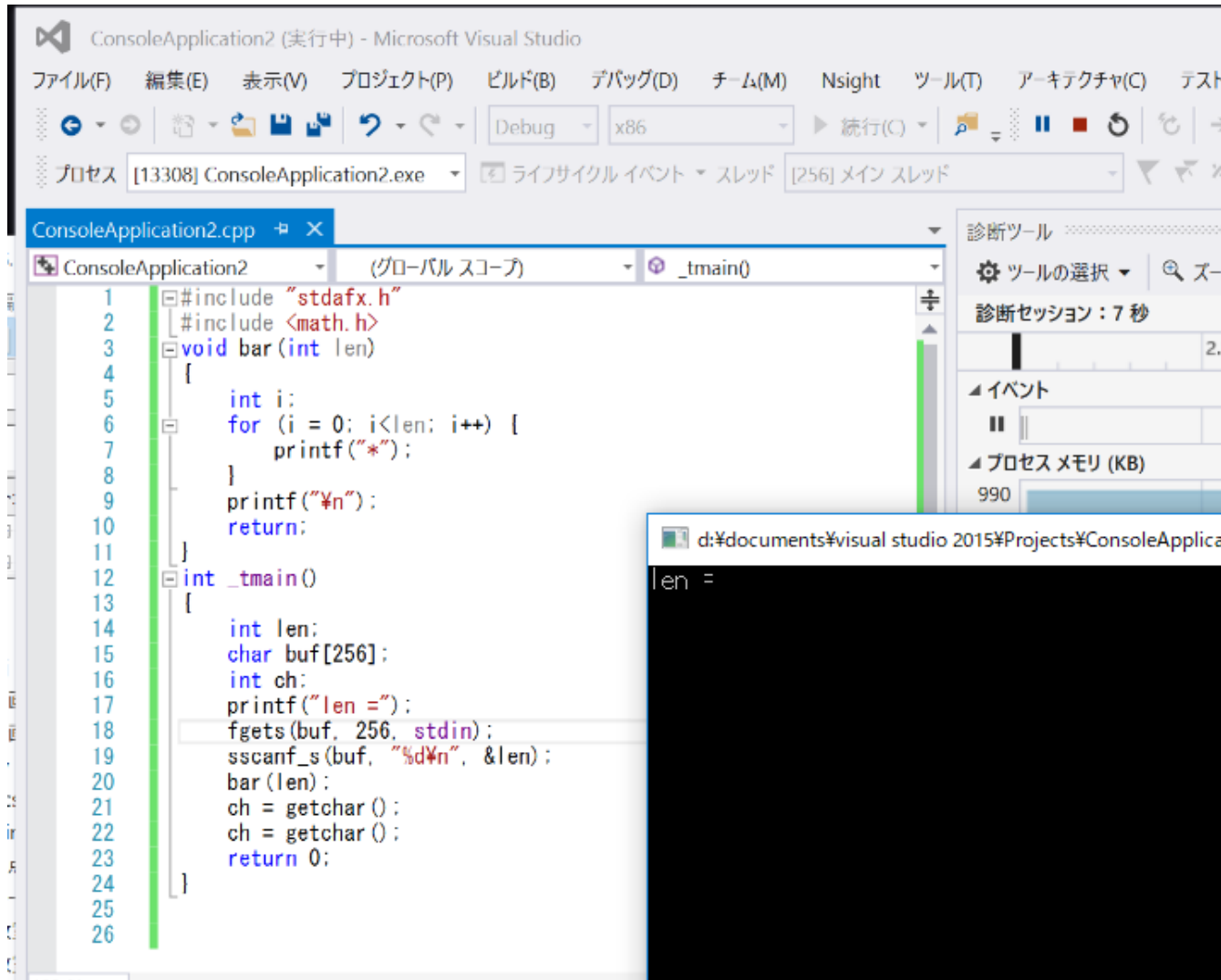
変数の値が変化する

名前	値	型
buf	0x0053f828 "濾w"	char[256]
ch	-16711613	int
len	6029369	int

呼び出し履歴

- 名前
- ConsoleApplication2.exe!
- [外部コード]
- [下のフレームは間違っている]

さらに「F10」キーを押すとステップ実行が続く



ConsoleApplication2 (実行中) - Microsoft Visual Studio

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) チーム(M) Nsight ツール(T) アーキテクチャ(C) テスト

プロセス [13308] ConsoleApplication2.exe ライフサイクル イベント スレッド [256] メイン スレッド

```
1 #include "stdafx.h"
2 #include <math.h>
3 void bar(int len)
4 {
5     int i;
6     for (i = 0; i < len; i++) {
7         printf("*");
8     }
9     printf("\n");
10    return;
11 }
12 int _tmain()
13 {
14     int len;
15     char buf[256];
16     int ch;
17     printf("len =");
18     fgets(buf, 256, stdin);
19     sscanf_s(buf, "%d\n", &len);
20     bar(len);
21     ch = getchar();
22     ch = getchar();
23     return 0;
24 }
25
26
```

診断ツール

ツールの選択

診断セッション: 7 秒

イベント

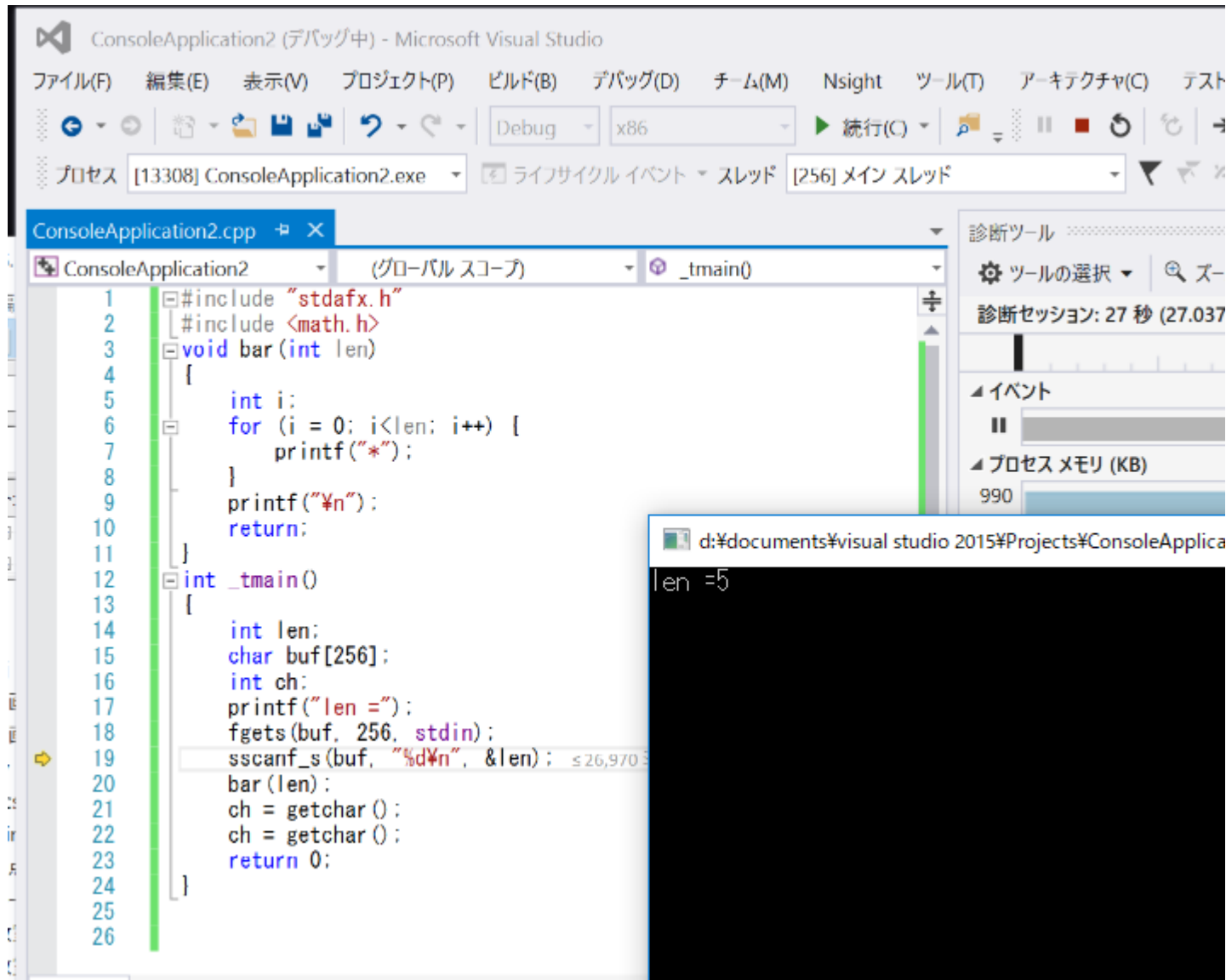
プロセス メモリ (KB)

990

d:\documents\visual studio 2015\Projects\ConsoleApplica

len =

さらに「**F10**」キーを押すとステップ実行が続く



The screenshot shows the Visual Studio IDE with a C++ project named 'ConsoleApplication2'. The code in 'ConsoleApplication2.cpp' is as follows:

```
1 #include "stdafx.h"
2 #include <math.h>
3 void bar(int len)
4 {
5     int i;
6     for (i = 0; i < len; i++) {
7         printf("*");
8     }
9     printf("\n");
10    return;
11 }
12 int _tmain()
13 {
14     int len;
15     char buf[256];
16     int ch;
17     printf("len =");
18     fgets(buf, 256, stdin);
19     sscanf_s(buf, "%d\n", &len);
20     bar(len);
21     ch = getchar();
22     ch = getchar();
23     return 0;
24 }
25
26
```

The execution window shows the output: 'len =5'. The 'Debug Console' on the right shows the process memory usage at 990 KB.

実行ウィンドウで「5 Enter」とすると実行が続く（5の部分は整数なら何でも良い）

ConsoleApplication2 (デバッグ中) - Microsoft Visual Studio

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) チーム(M) Nsight ツール(T) アーキテクチャ(C) テスト

Debug x86 実行(C) [13308] ConsoleApplication2.exe ライフサイクル イベント スレッド [256] メイン スレッド

```
1 #include "stdafx.h"
2 #include <math.h>
3 void bar(int len)
4 {
5     int i;
6     for (i = 0; i < len; i++) {
7         printf("*");
8     }
9     printf("\n");
10    return;
11 }
12 int _tmain()
13 {
14     int len;
15     char buf[256];
16     int ch;
17     printf("len =");
18     fgets(buf, 256, stdin);
19     sscanf_s(buf, "%d\n", &len);
20     bar(len);
21     ch = getchar(); <1 ミリ秒経過
22     ch = getchar();
23     return 0;
24 }
25
26
```

診断ツール
ツールの選択
診断セッション: 27 秒 (27.037)
27.035秒
イベント
プロセス メモリ (KB)
990

d:\documents\visual studio 2015\Projects\ConsoleApplica
len =5

さらに「F10」キーを押すとステップ実行が続く

例題 3 (b) . 棒グラフでのステップイン



- ステップ実行とステップインを行う
 - ステップ実行 (「F10」キー)
 - ステップイン (「F11」キー)

- 例題 2 のプログラムをそのまま使う

プログラム実行順



- 普通, プログラム中の文は, 上から下へ順に実行される
- 関数呼び出しでは, 関数の先頭に「ジャンプ」する.

このことは, C言語が「手続き型言語」と言われる理由の1つ
- 関数呼び出しの例 `bar(len);`
- 呼び出された関数の中で `return` 文に出会うと, 関数呼び出しの場所に戻る.

プログラムの流れ



プログラムの実行開始

↓
メイン関数

関数呼び出し

bar(len);

↓
プログラムの実行終了

```
#include "stdio.h"
#include <math.h>
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
        printf("*");
    }
    printf("¥n");
    return;
}

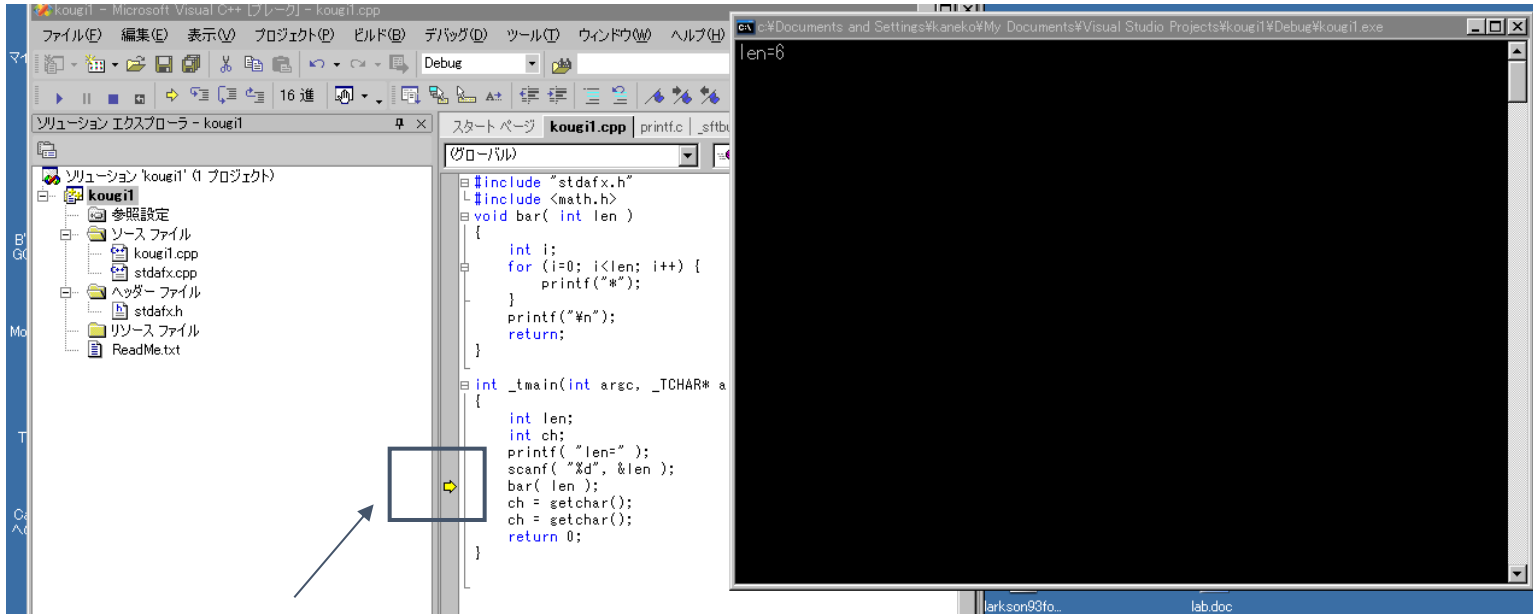
int main()
{
    int len;
    int ch;
    printf( "len=" );
    scanf( "%d", &len );
    bar( len );
    ch = getchar();
    ch = getchar();
    return 0;
}
```

→ bar 関数

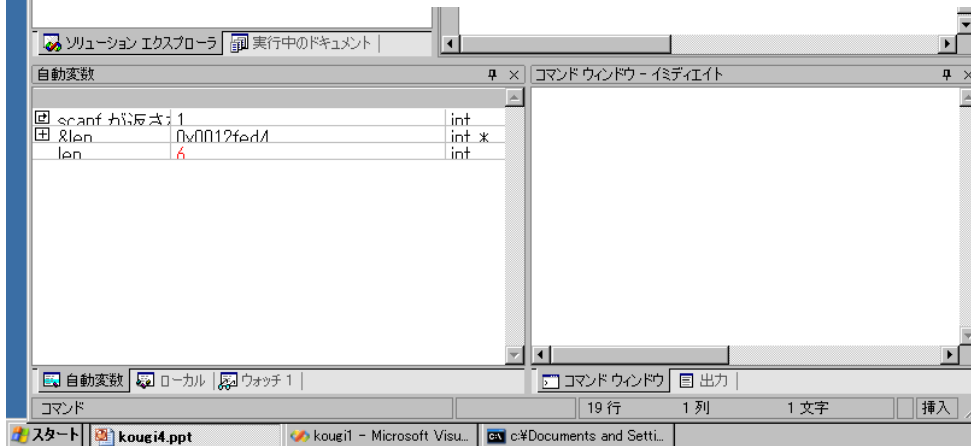
return;

* printf, scanf の
呼び出しについては
図では省略 • 50

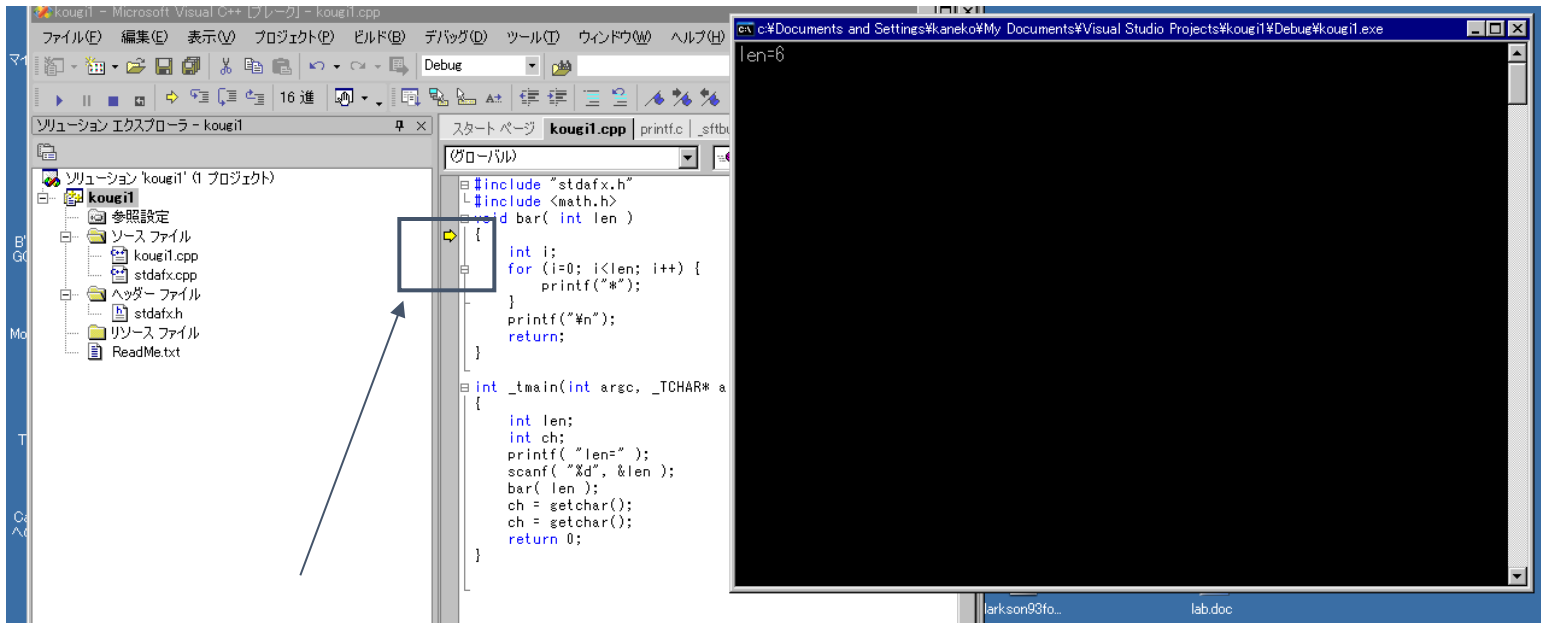
ステップイン機能



マークが「**bar(len);**」の行に来たときに
F11 キーを押して、ステップイン機能を使う



ステップイン機能



マークが **bar** 関数の内部に移る
(後は, 「F10」キーでステップ実行を続ける)



補足説明事項

ブレークポイント



- Microsoft Visual Studio C++ のブレークポイント機能を試す
 - ブレークポイント

Microsoft Visual Studio.NET でのブレークポイント設定

A screenshot of the Microsoft Visual Studio IDE. The title bar reads "ConsoleApplication2 - Microsoft Visual Studio". The menu bar includes "ファイル(F)", "編集(E)", "表示(V)", "プロジェクト(P)", "ビルド(B)", and "デバッグ(D)". The toolbar shows various icons for file operations and debugging, with a "Debug" dropdown menu and "x86" architecture selected. The main editor window shows a C++ file named "ConsoleApplication2.cpp" with the following code:

```
10     return:  
11 }  
12 int _tmain()  
13 {  
14     int len;  
15     char buf[256];  
16     int ch;  
17     printf("len =");  
18     fgets(buf, 256, stdin);  
19     sscanf_s(buf, "%d\n", &len);  
20     bar(len);  
21     ch = getchar();  
22     ch = getchar();
```

A red dot, representing a breakpoint, is placed on the left margin of line 20. A white box highlights this red dot, with a black arrow pointing from the text box below to it. The left sidebar shows the "Solution Explorer" and "Server Explorer" tabs.

ここをクリックして、ブレークポイントの設定
「●」はブレークポイント設定済みの印

ブレークポイントでプログラム実行が中断するので、
実行結果の画面が消えずに残る