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

CC BY NC SA

C言語プログラムの実行順序の理解、Microsoft Visual Studio C++のステップ実行機能を用いた変数値変化の観察。

【学習内容の構成】

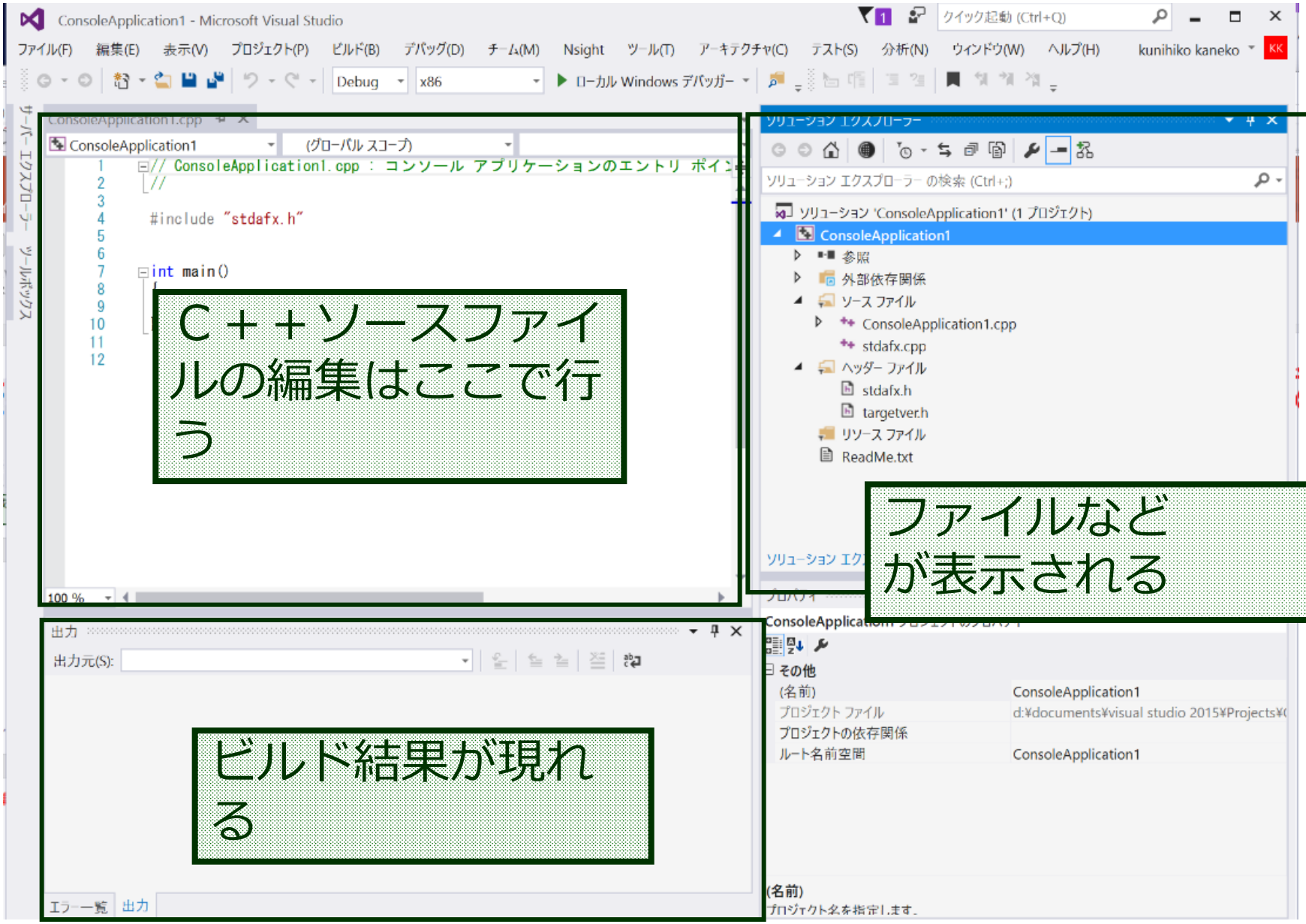
1. **プログラムの順次実行**：メイン関数から始まり上から下へ順に実行される基本原理
2. **ビルドと実行ファイル生成**：C++ソースファイルからの実行ファイル生成、構文エラー時の対処
3. **条件分岐**：if文・else文による条件式の成否に応じた処理の切り替え
4. **関数呼び出しと戻り**：関数呼び出しによるジャンプとreturn文による呼び出し元への復帰
5. **デバッグ機能**：ステップ実行（F10）、ステップイン（F11）、ブレークポイントによる実行制御と変数観察

- 前提：Microsoft Visual Studio C++の基本操作、C言語の変数・繰り返し・関数の基礎知識
- 意義：プログラムの動作原理の理解、デバッグ技法の習得

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



```
#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:¥¥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;
}
```

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

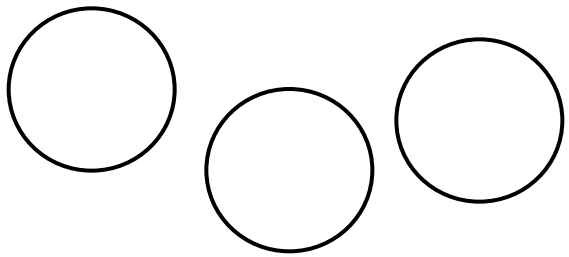
20回の繰り返し (i = 0, 1, ... 19)

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

ビルド

Microsoft Visual Studio C++

実行ファイル

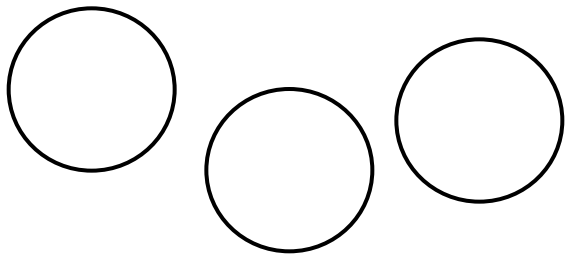
その他，ビルドの結果として生成されるファイル

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）

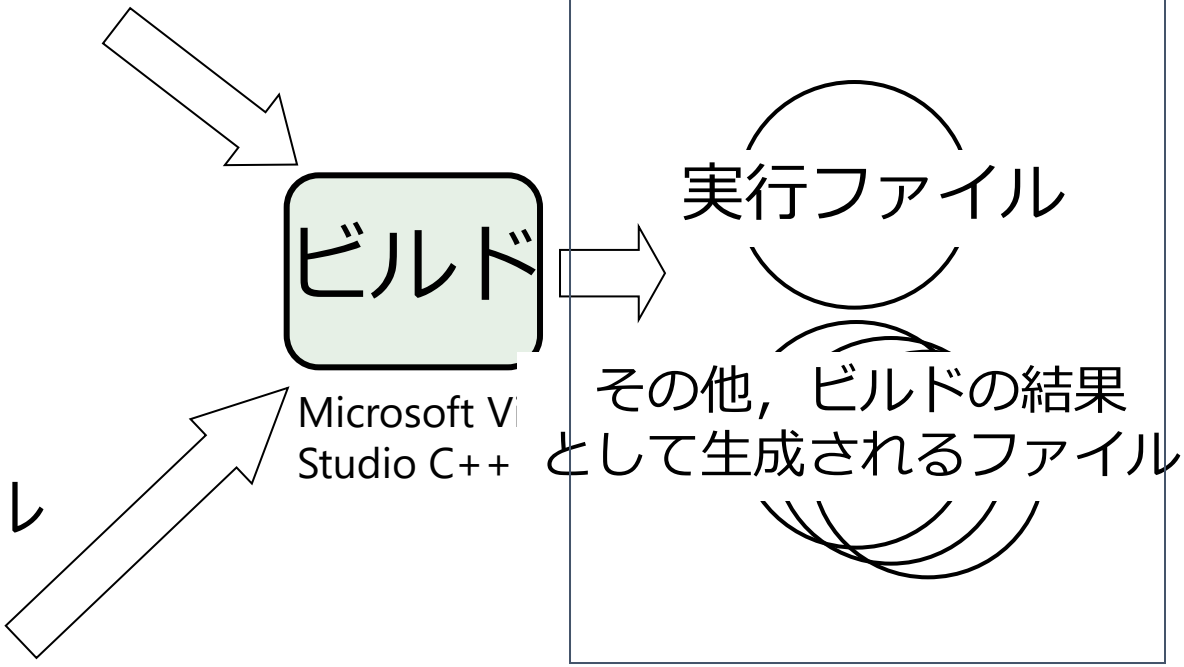
「消去（clean）」の操作

```
#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

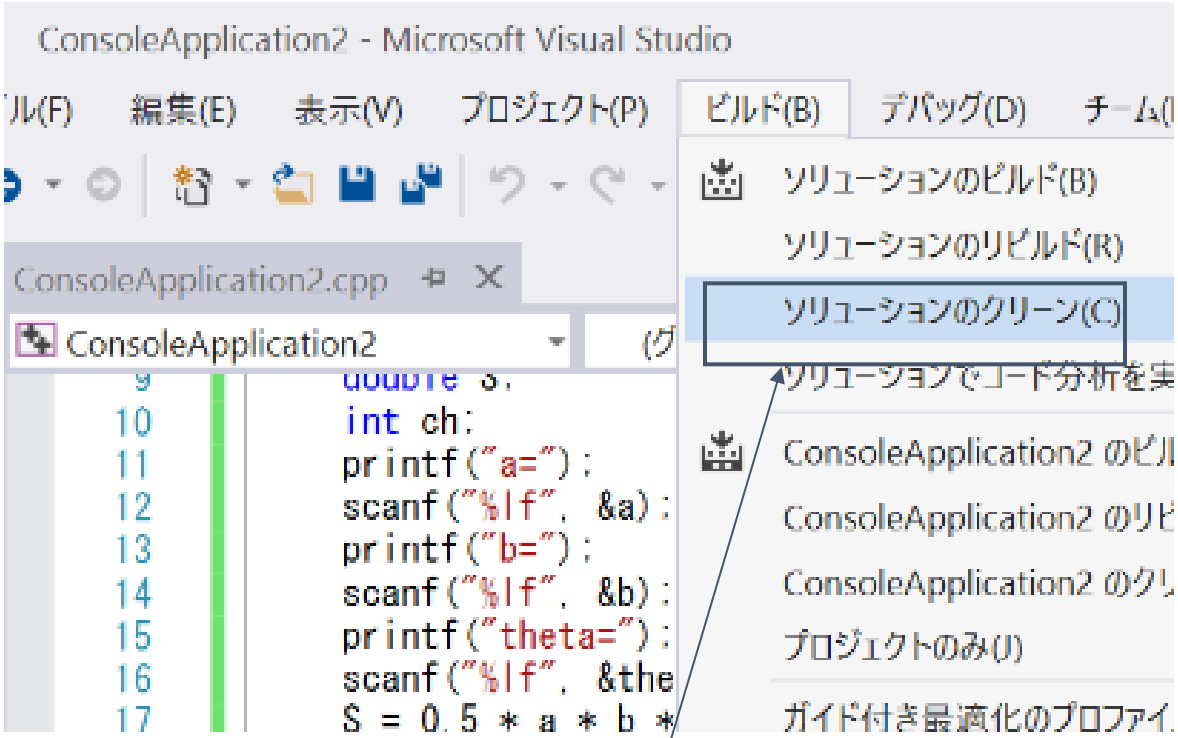


その他のファイル



「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）

「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無い（ディスク使用量の節約になる）

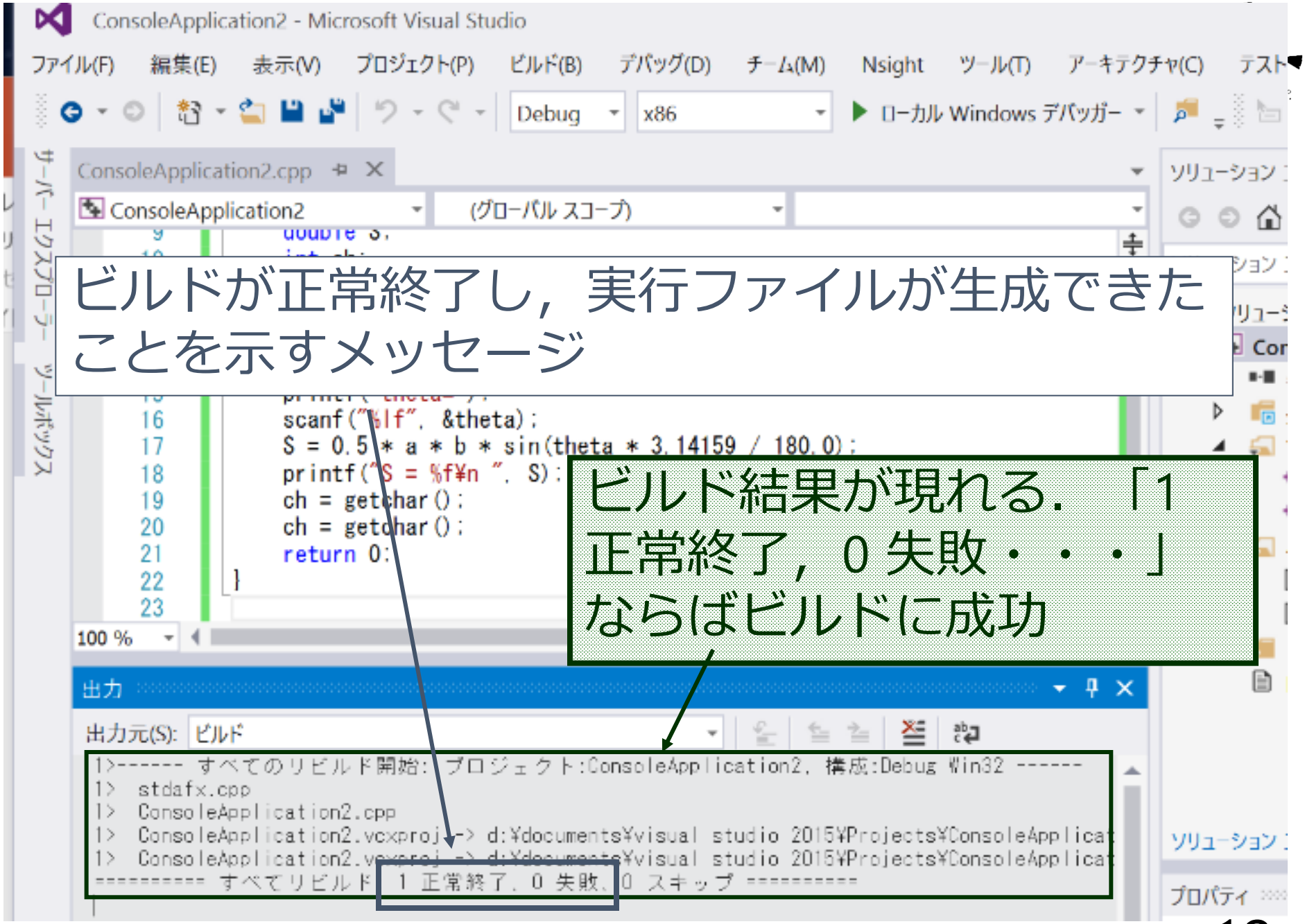
構文エラーの例

- 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）

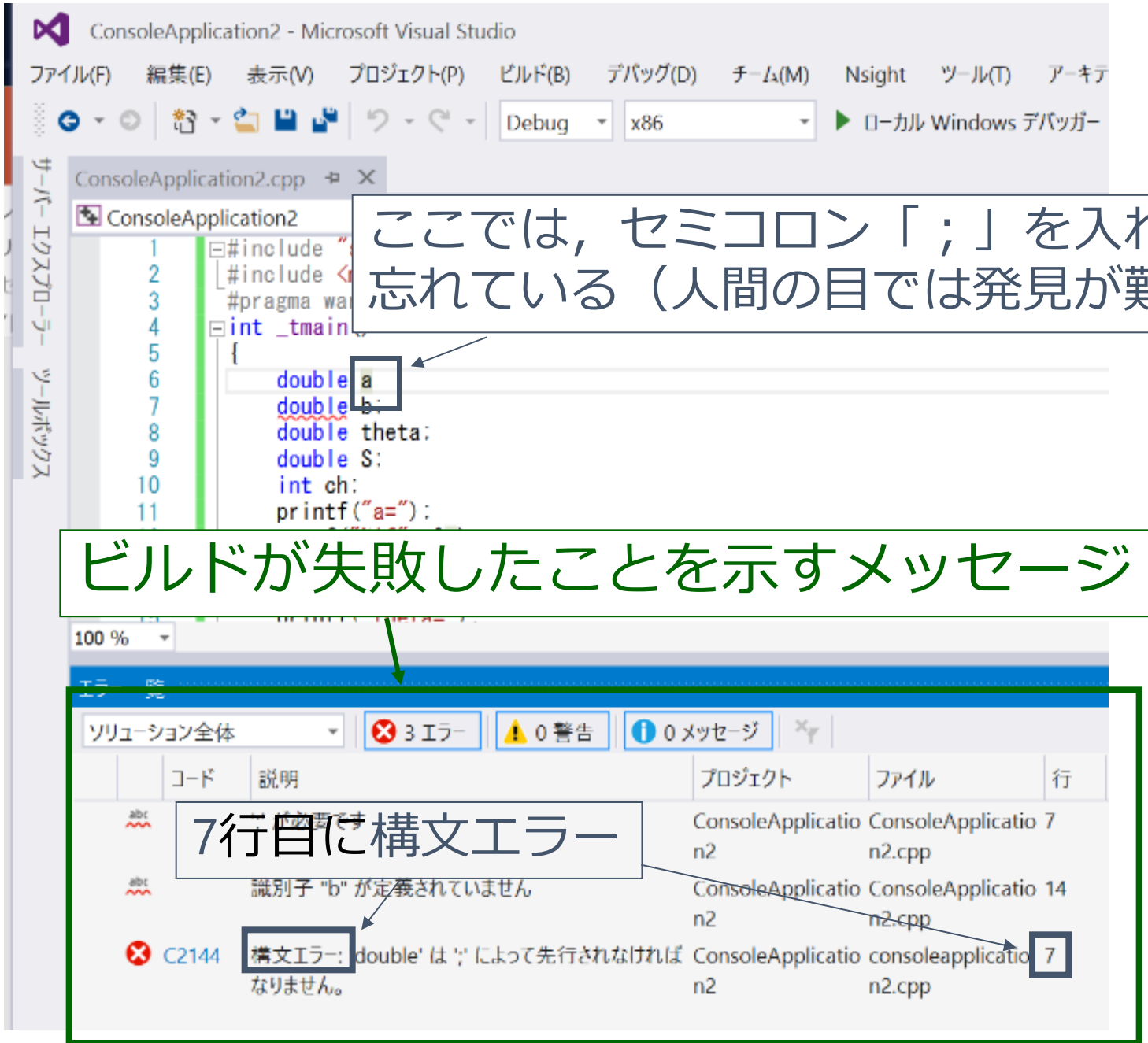
- 「"」　抜け

- 「;」　忘れ

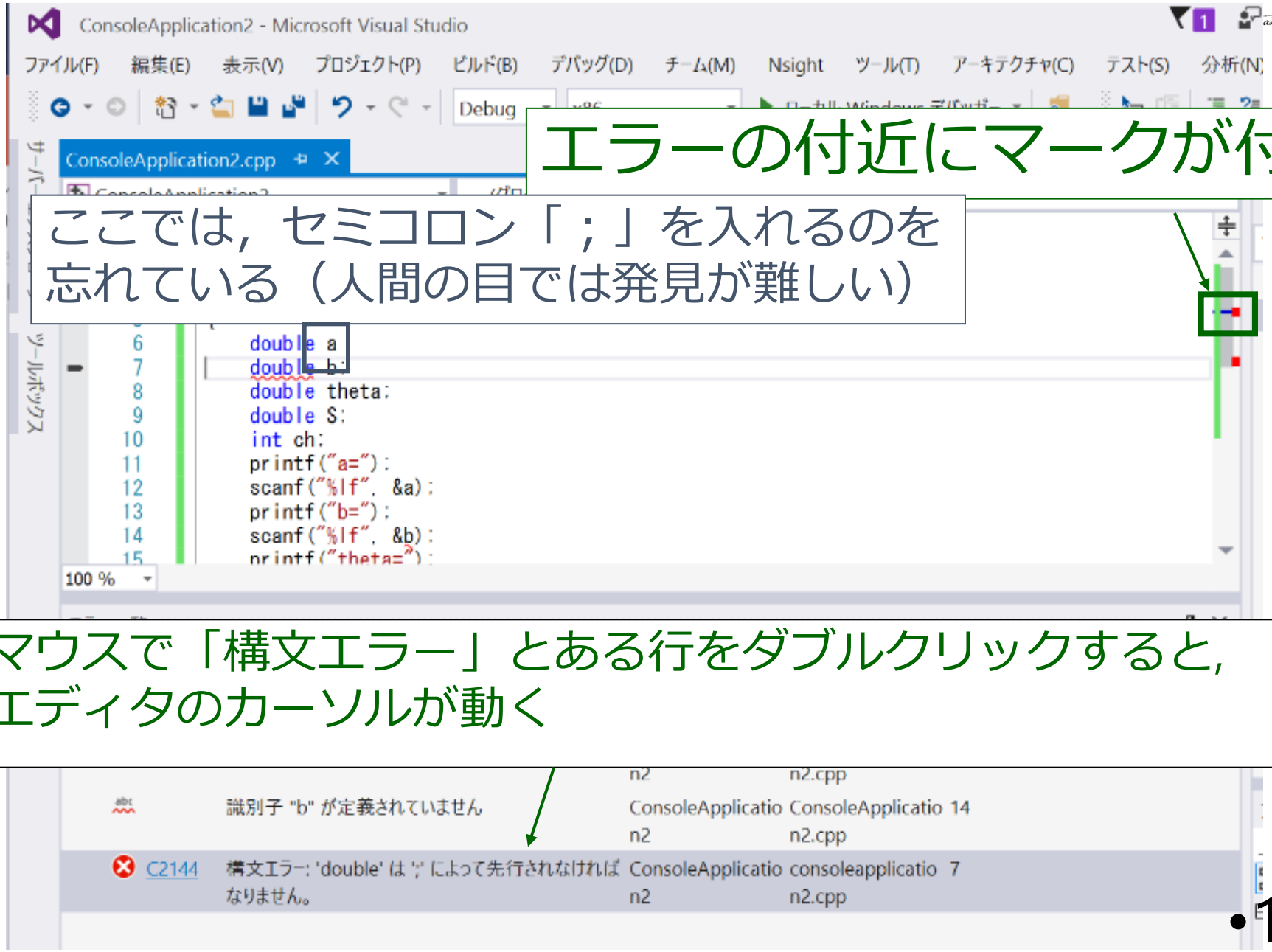
- など


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
Debug
x86
ローカル Windows デバッガー
ConsoleApplication2.cpp
ConsoleApplication2
(グローバル スコープ)
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
scanf("%lf", &theta);
S = 0.5 * a * b * sin(theta * 3.14159 / 180.0);
printf("S = %f¥n ", S);
ch = getchar();
ch = getchar();
return 0;
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¥ConsoleApplicat
1> ConsoleApplication2.vcxproj -> d:¥documents¥visual studio 2015¥Projects¥ConsoleApplicat
========== すべてリビルド: 1 正常終了, 0 失敗, 0 スキップ ==========
ソリューション
プロパティ

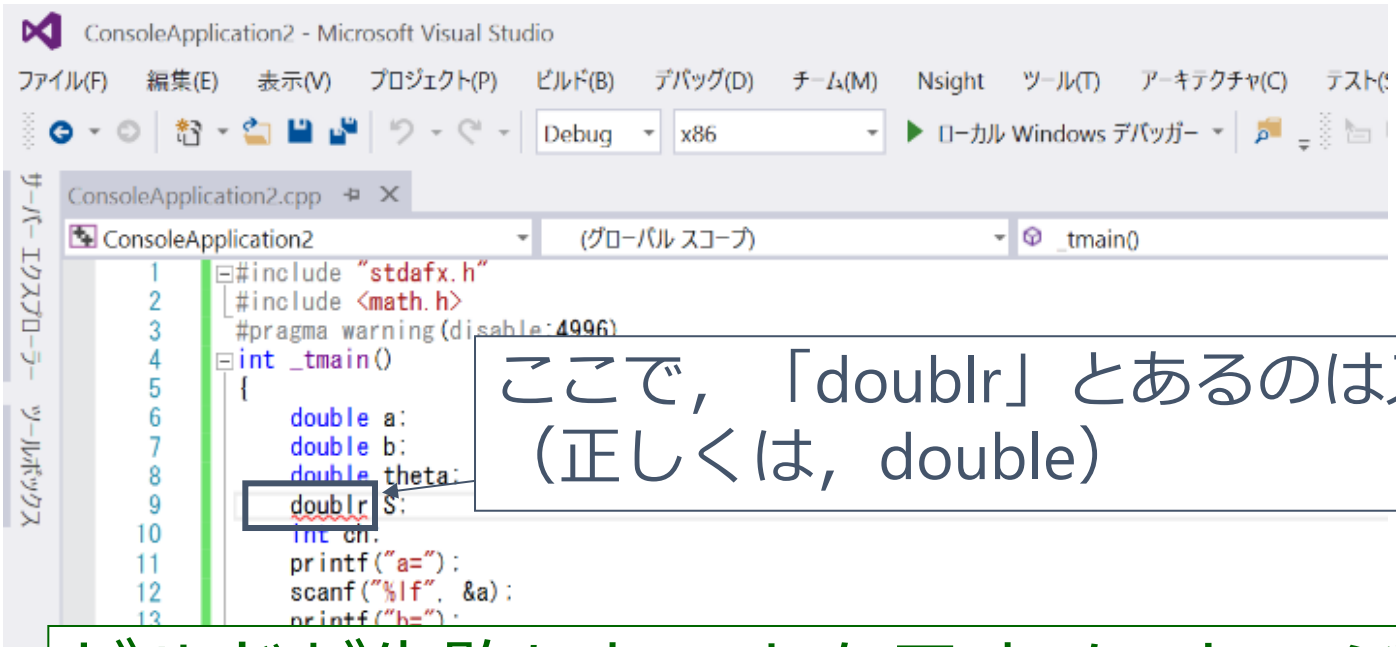
構文エラーの例（1）


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)
Debug
x86
ローカル Windows デバッガー
ConsoleApplication2.cpp
ConsoleApplication2
1 #include
2 #include
3 #pragma
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る（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
ConsoleApplication2
ConsoleApplication2.cpp
7
識別子 "b" が定義されていません
ConsoleApplication2
ConsoleApplication2.cpp
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。
ConsoleApplication2
consoleapplication2.cpp
7
7行目に構文エラー

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) アーキテクチャ(C) テスト(S) 分析(N)
Debug
ConsoleApplication2.cpp
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）
6 double a
7 double b
8 double theta;
9 double S;
10 int ch;
11 printf("a=");
12 scanf("%lf", &a);
13 printf("b=");
14 scanf("%lf", &b);
100 %
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
識別子 "b" が定義されていません
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。

構文エラーの例（2）



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）

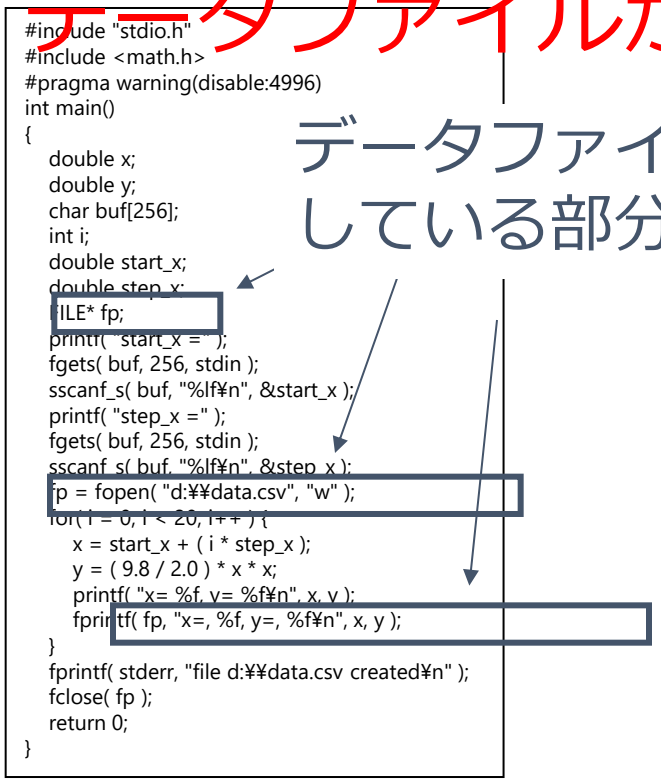
	コード	説明	プロジェクト	ファイル	行	抑制状態
		識別子 "doublr" が定義されていません	ConsoleApplicatio n2	ConsoleApplicatio n2.cpp	9	
⊗	C2065	'doublr': 定義されていない識別子です。	ConsoleApplicatio n2	consoleapplicatio n2.cpp	9	
⊗	C2146	構文エラー: ';' が、識別子 'S' の前に必要です。	ConsoleApplicatio n2	consoleapplicatio n2.cpp	9	
⊗	C2065	'S': 定義されていない識別子です。	ConsoleApplicatio	consoleapplicatio	9	
⊗	C2065	'S': 定義されてい		n2.cpp		
⊗	C2065	'S': 定義されていない識別子です。	ConsoleApplicatio n2	consoleapplicatio n2.cpp	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 );
    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;
}
```

データファイルに関係している部分

C++ソースファイル

ビルド

実行ファイル

データファイル

その他のファイル

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;
}
```

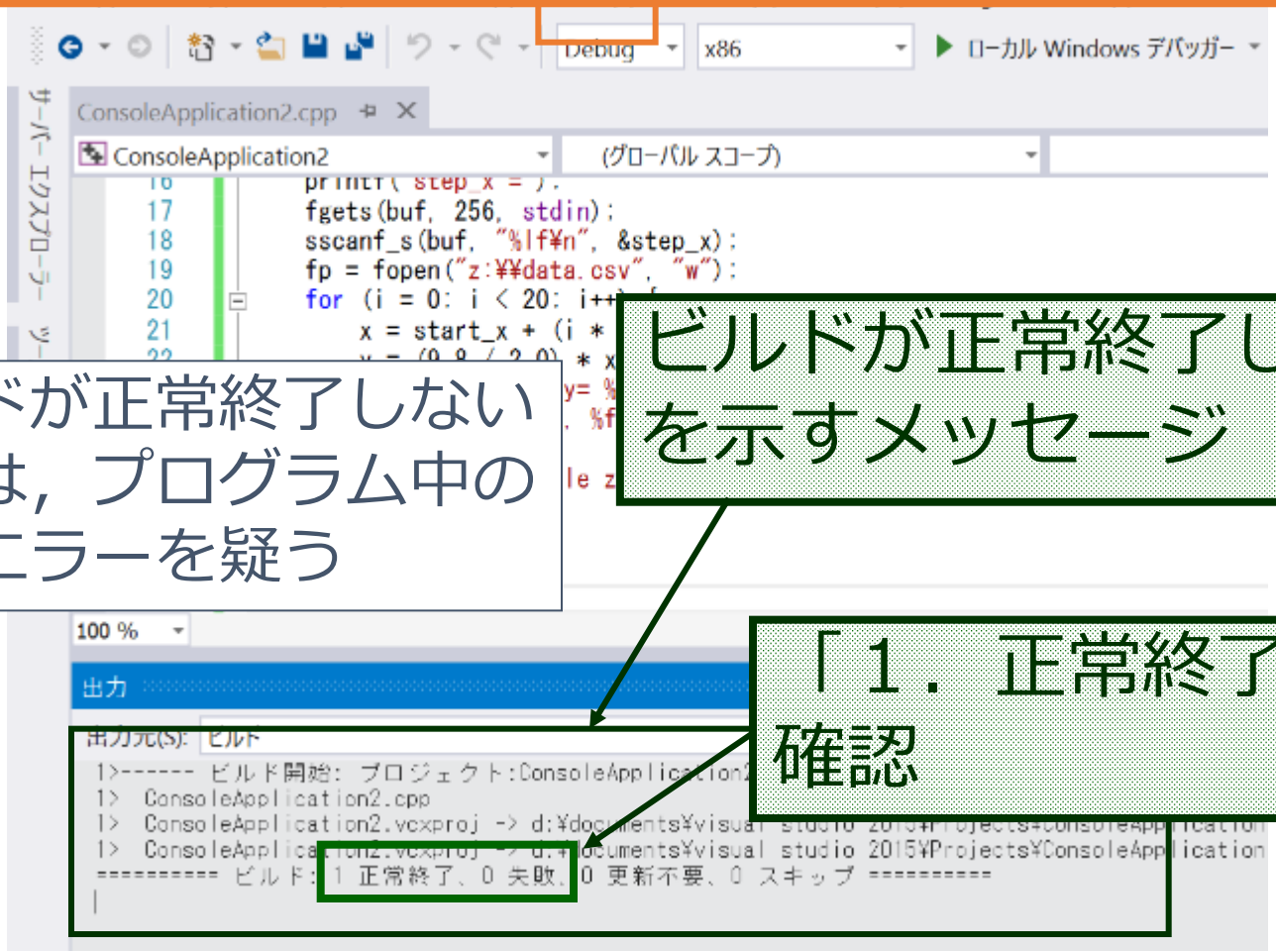
データファイル名

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

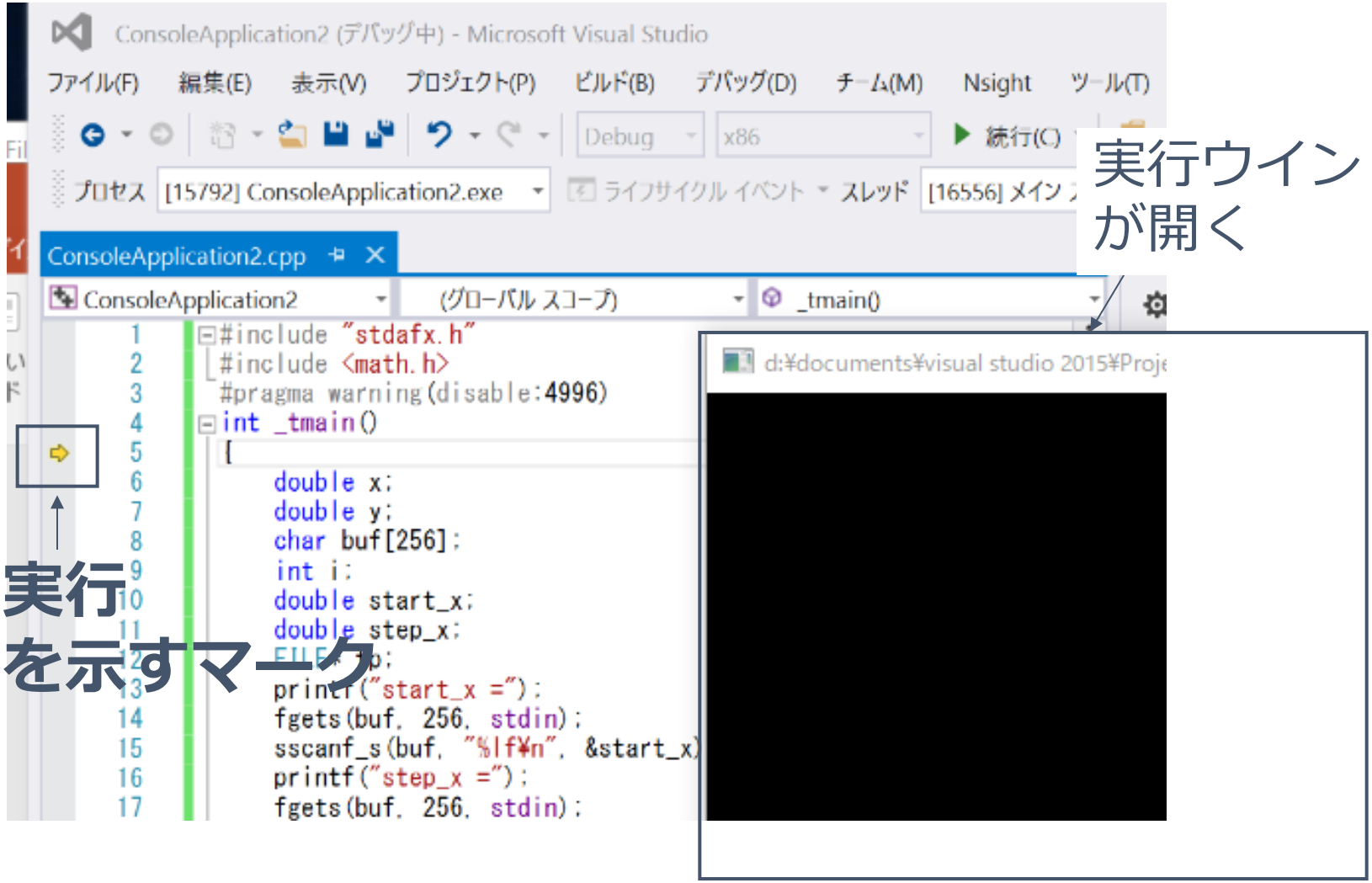
ビルドの手順：
「ビルド」→「ソリューションのビルド」



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

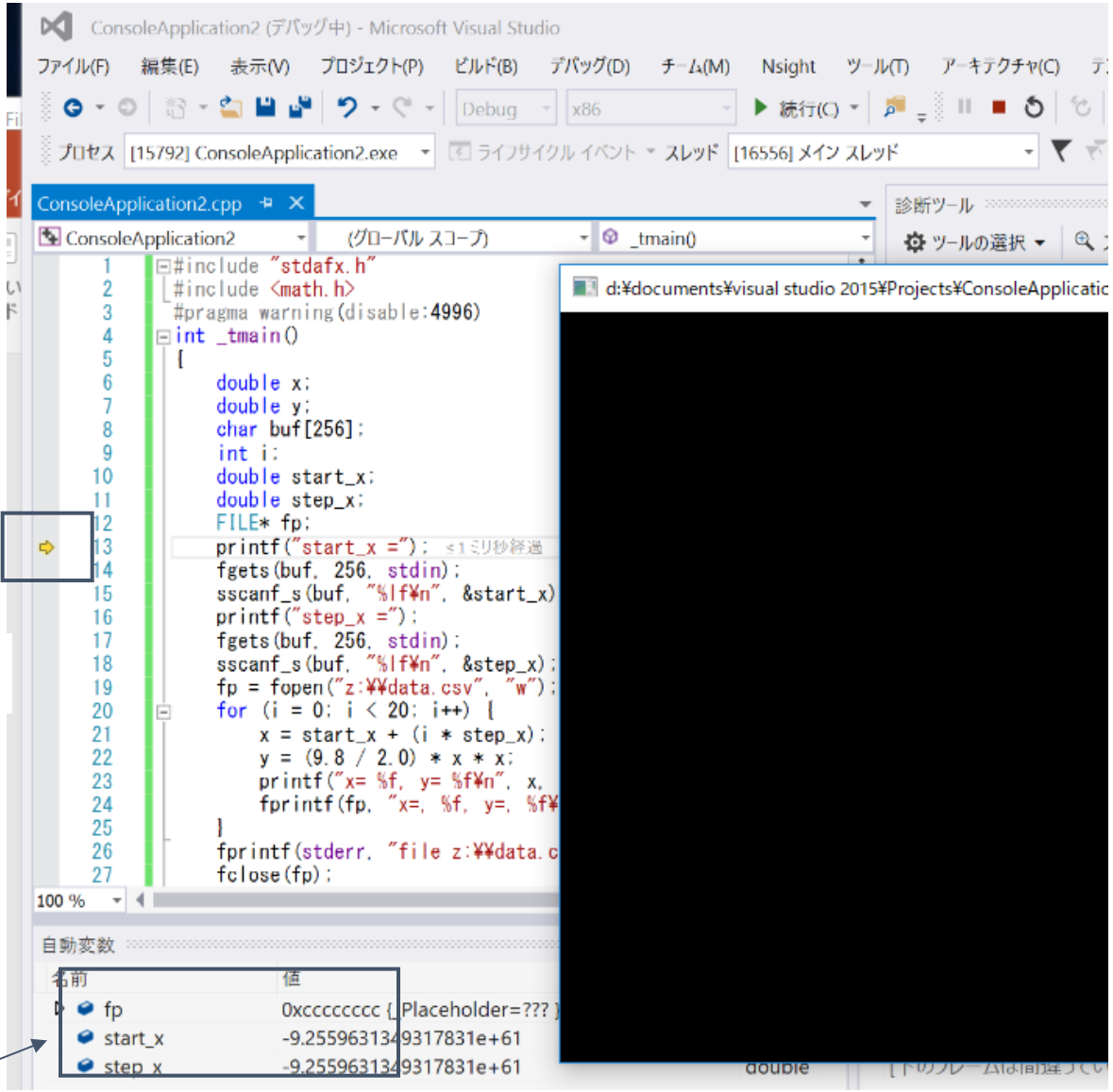
「１．正常終了」を確認



実行ウインドウが開く

実行を示すマーク

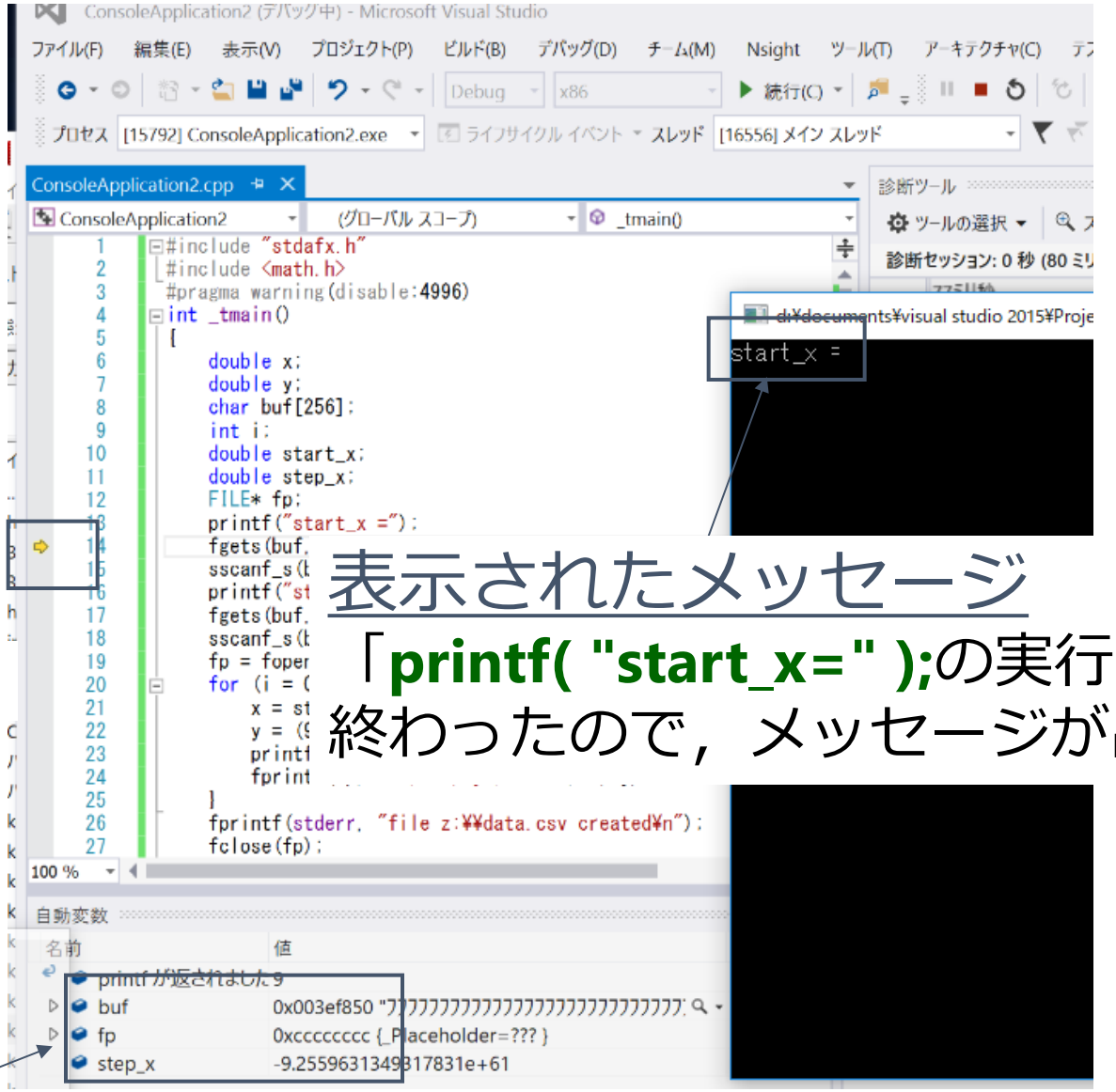
「**F10**」キーを押すとステップ実行が始まる．（マウスカーソルは，Microsoft Visual Studio C++ 内に入れておく）



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



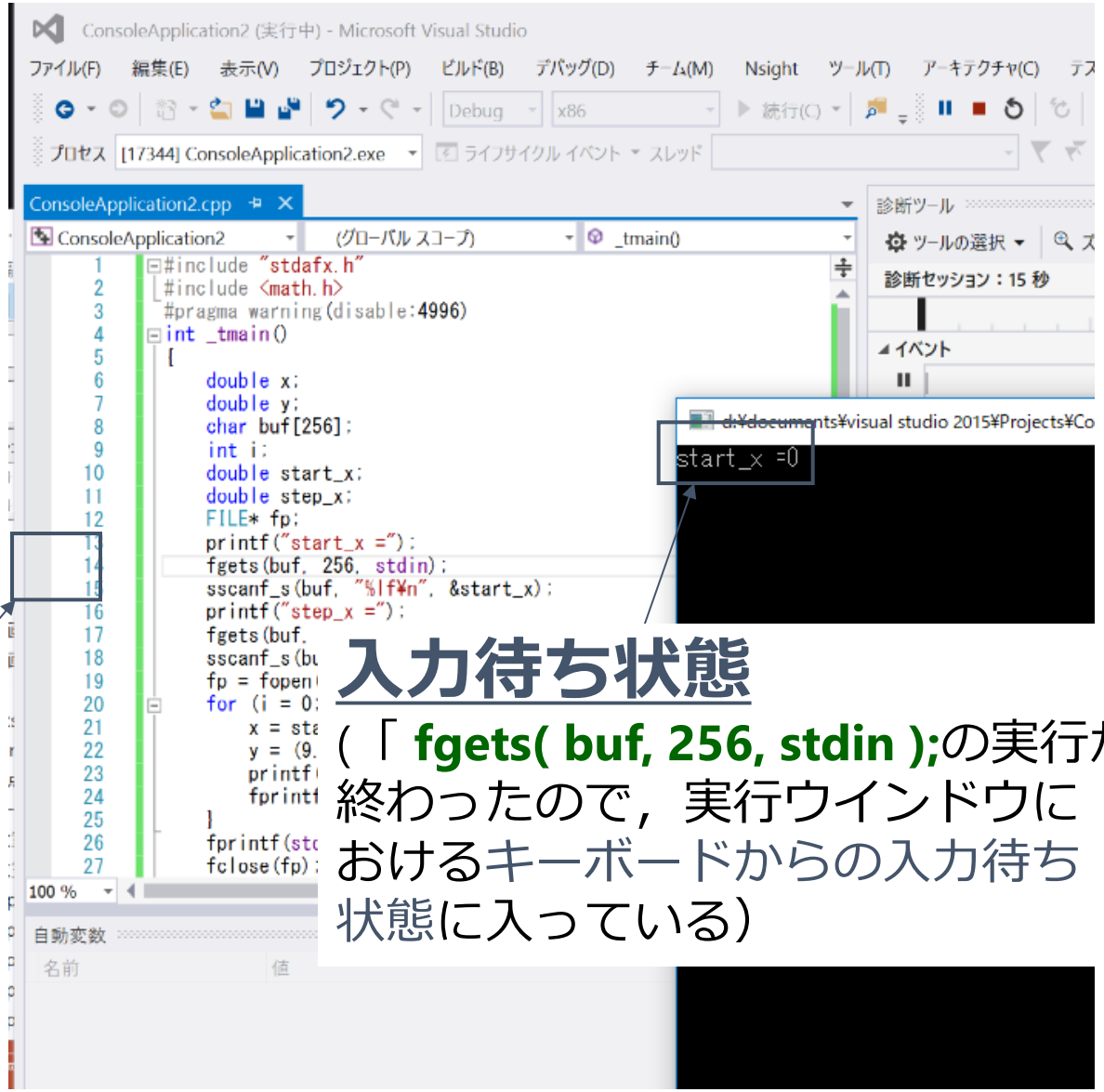
マークが進む

変数の値を観察できる

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

さらに「**F10**」キーを押すとステップ実行が続く

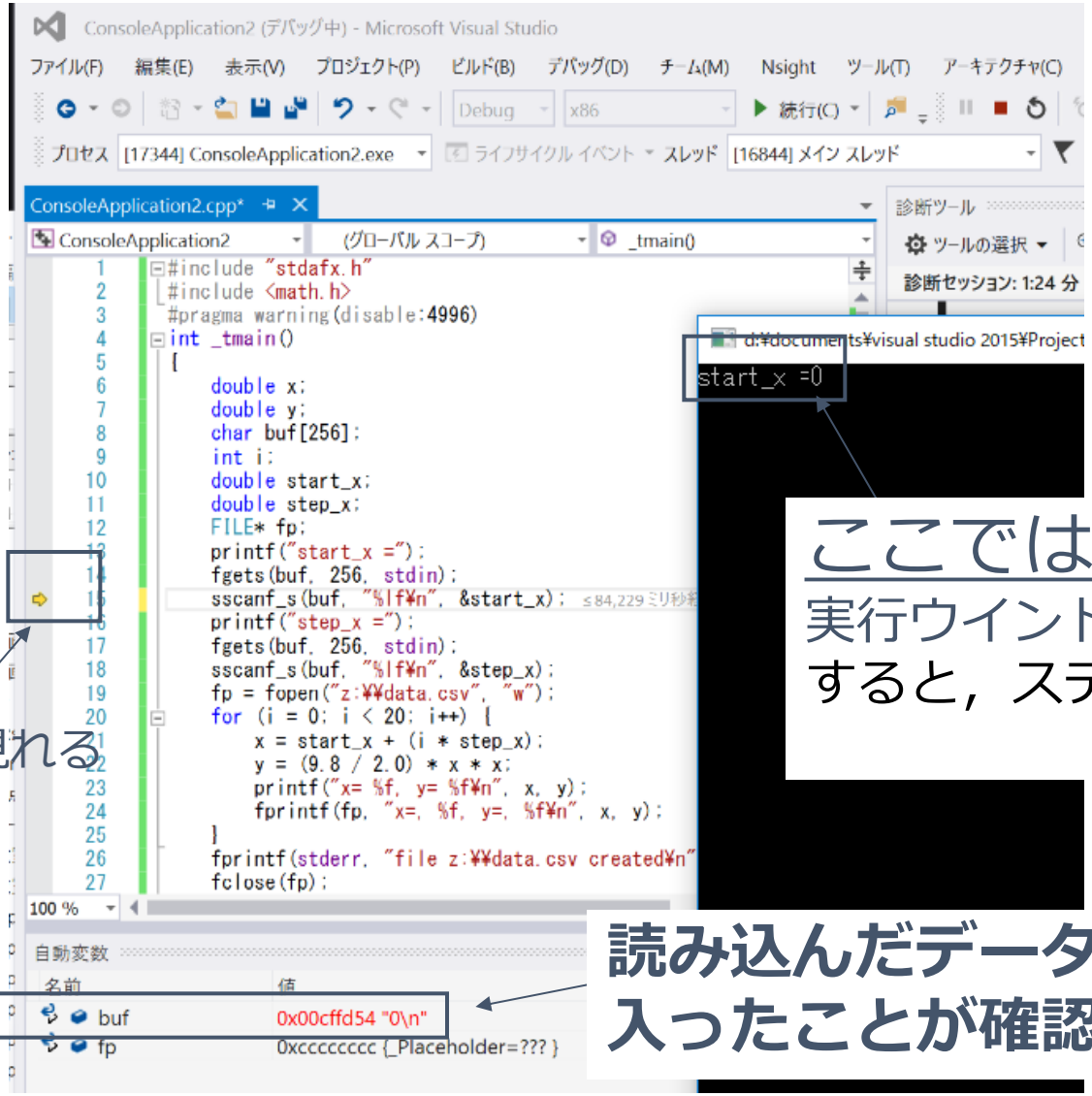


マークが変化

入力待ち状態

(「 **fgets(buf, 256, stdin);**の実行が終わったので，実行ウイ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）



マークが現れる

ここでは　0 Enter

実行ウインドウで，「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

実行ウインドウで「**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，新しいウインドウが開く
- 新しいウイ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

- ウインドウは消えるが，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，新しいウインドウが開く

  「F10」は，Microsoft Visual Studio C++のウイ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

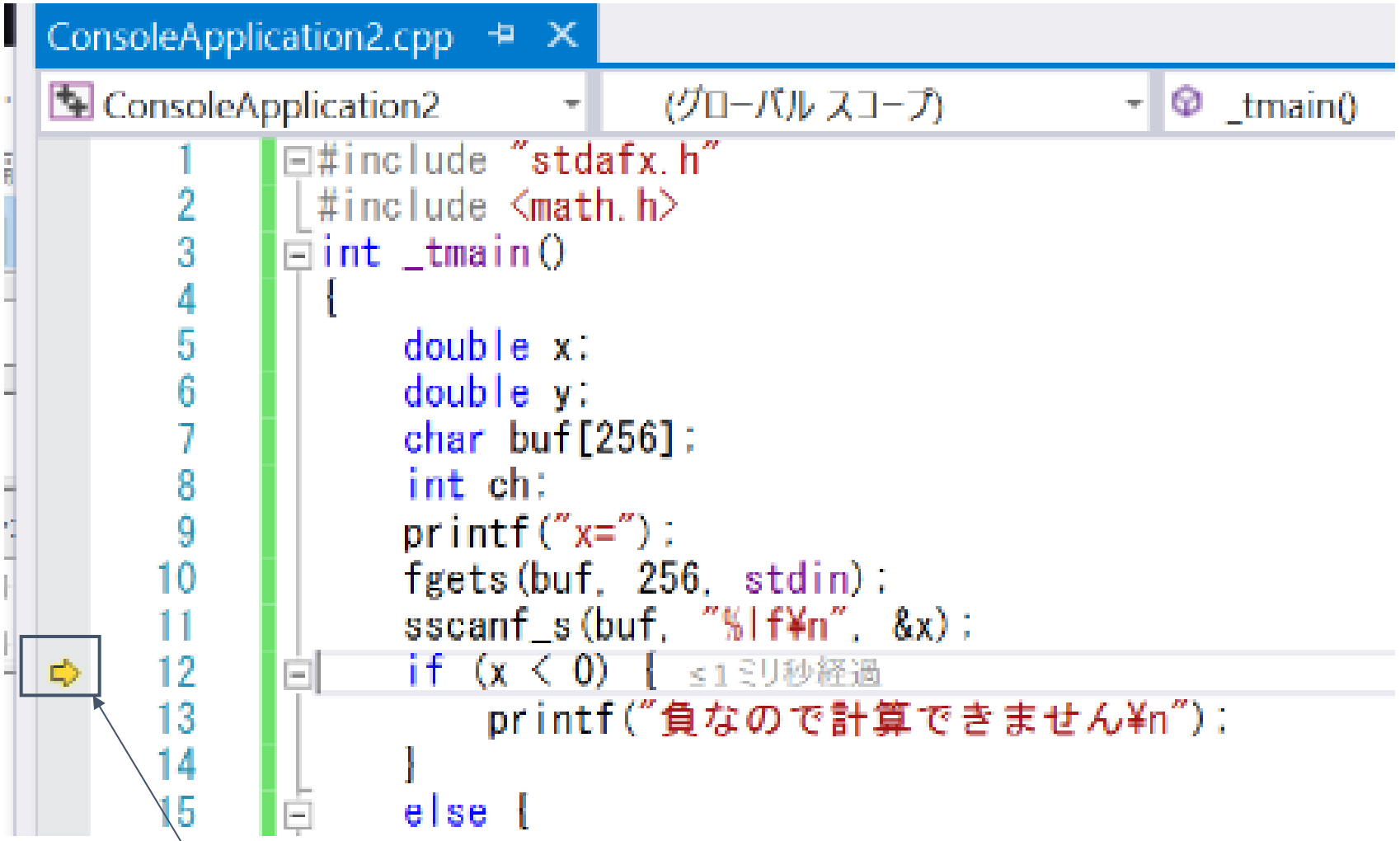
実行順
(x<0) の場合

```
#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;
}
```

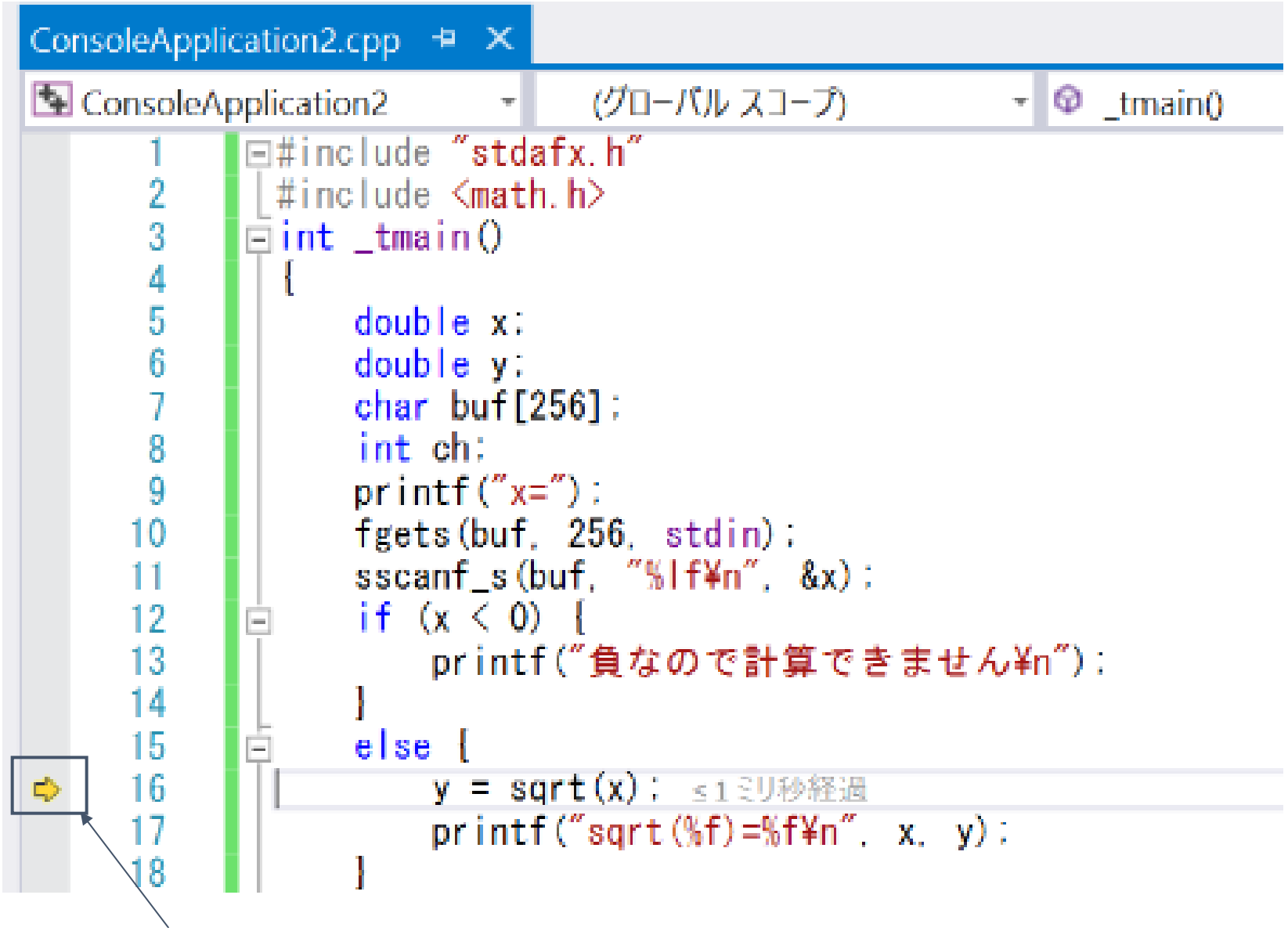
実行順
(x≧0) の場合

```
#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



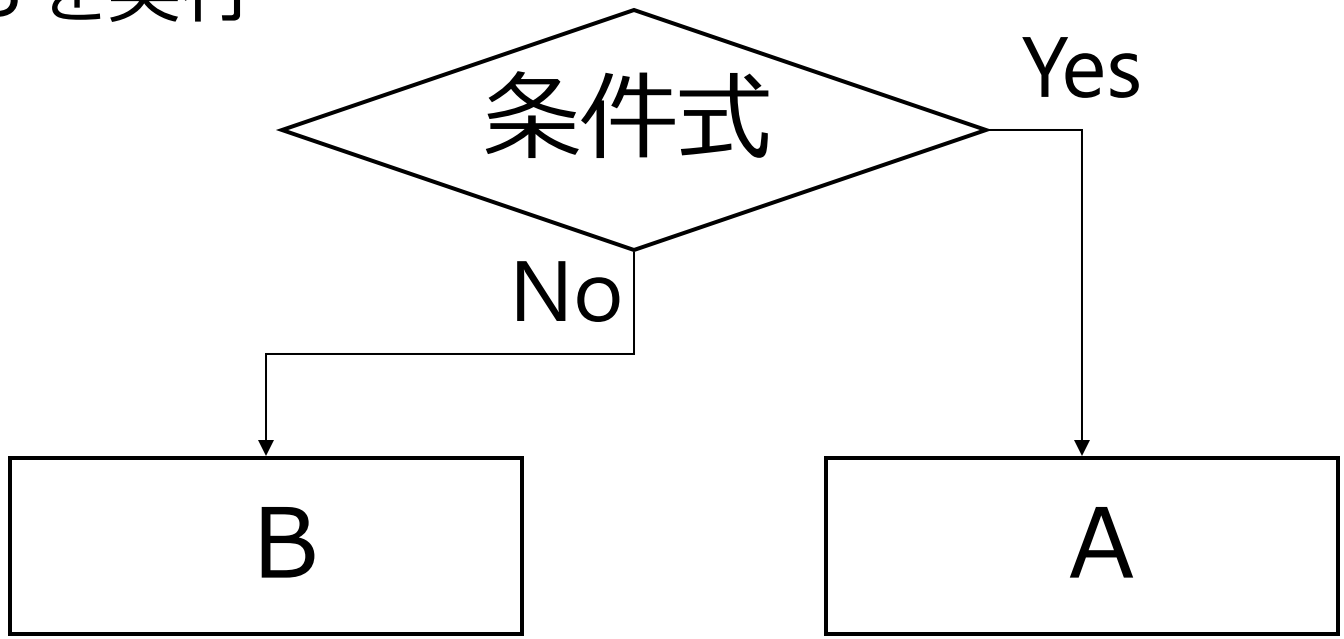
マークが「if (x < 0) {」に来たときに，
「F10」を押すと



「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



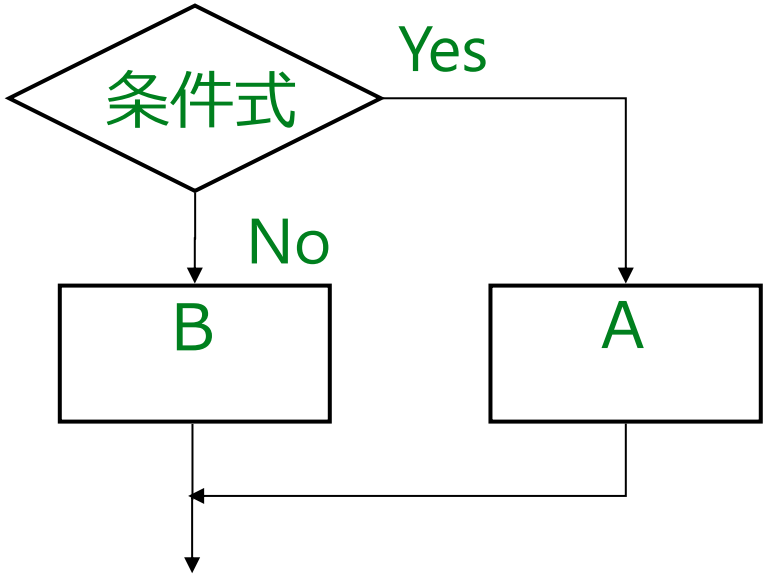
if 文と else 文

「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;
  … ;
}
else {
  式;
  … ;
}
```

A

B



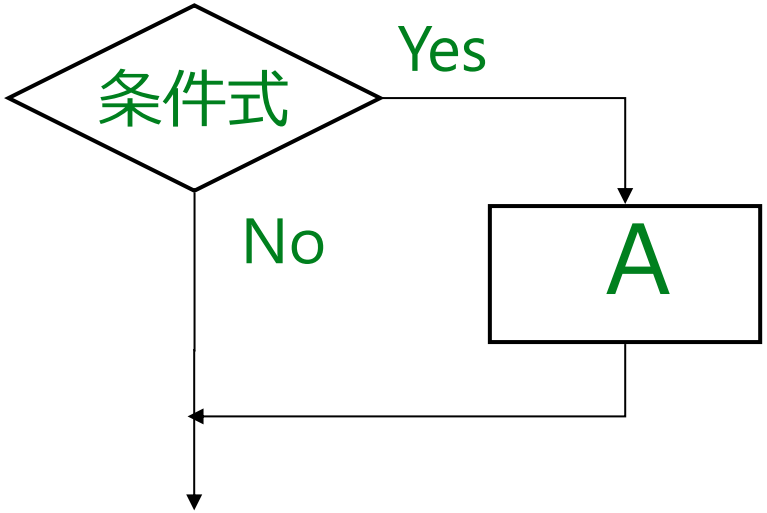
if 文

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



比較演算

- 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
- 字下げを忘れると
  - プログラムは動くが，読みづらい

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;
}
```

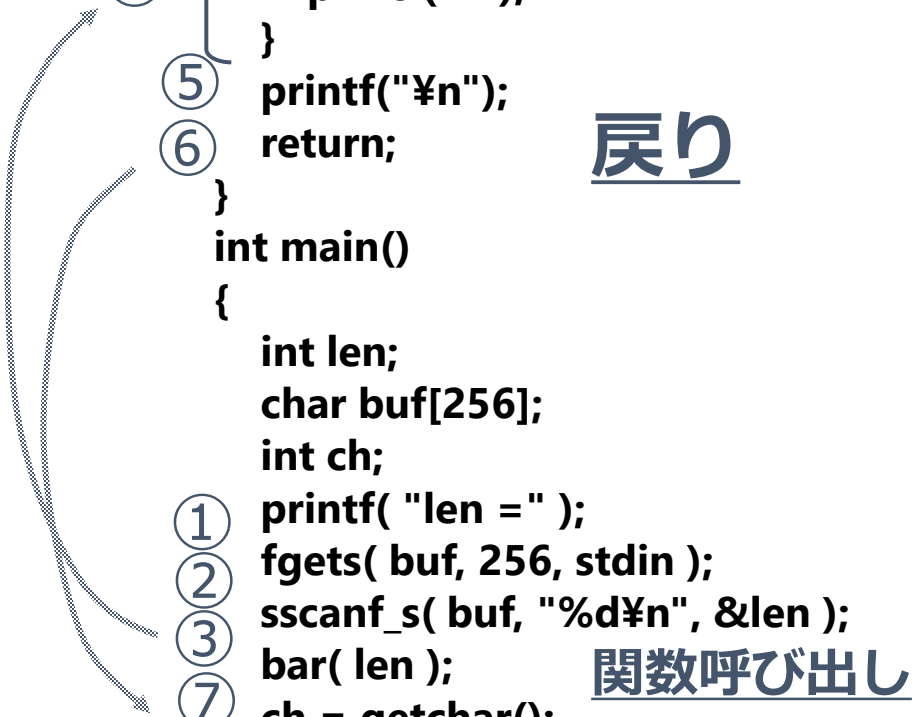
bar関数

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



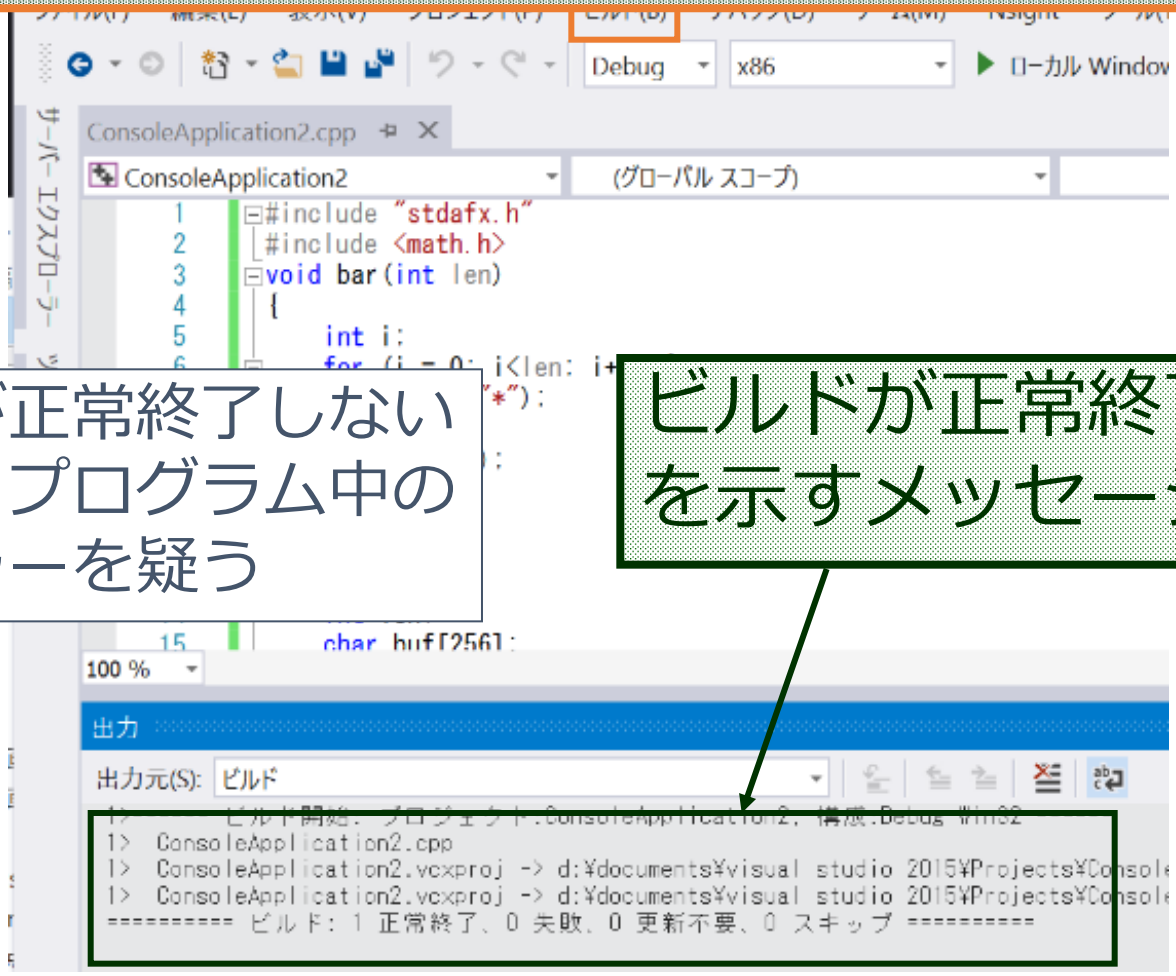
```
#include "stdio.h"
#include <math.h>
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
④       printf("*");
    }
⑤   printf("¥n");
⑥   return;
}
int main()
{
    int len;
    char buf[256];
    int ch;
①   printf( "len =" );
②   fgets( buf, 256, stdin );
③   sscanf_s( buf, "%d¥n", &len );
    bar( len );
⑦   ch = getchar();
⑧   ch = getchar();
⑨   return 0;
}
```


メイン関数の先頭行
がプログラム実行の始まり

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

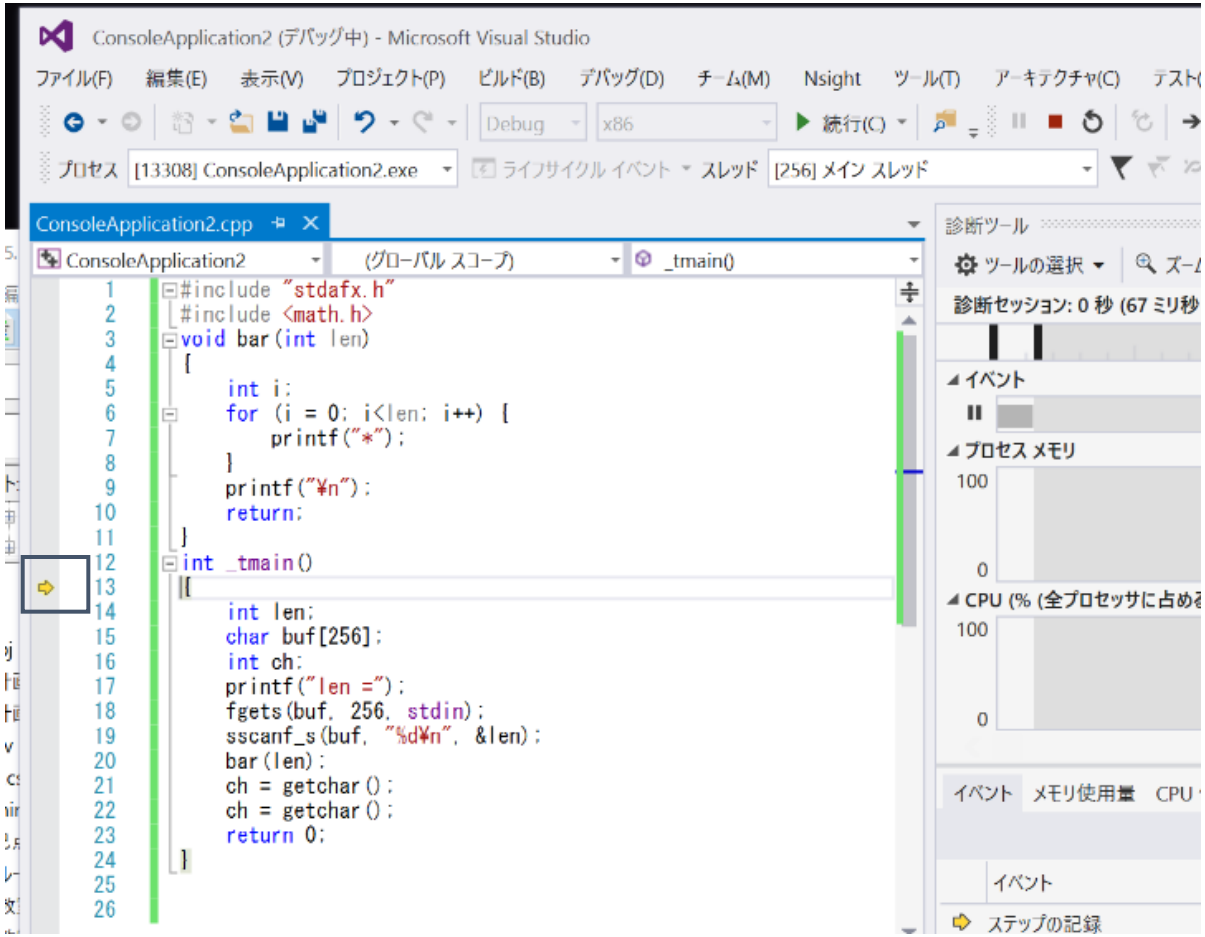
ビルドの手順：
「ビルド」→「ソリューションのビルド」



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

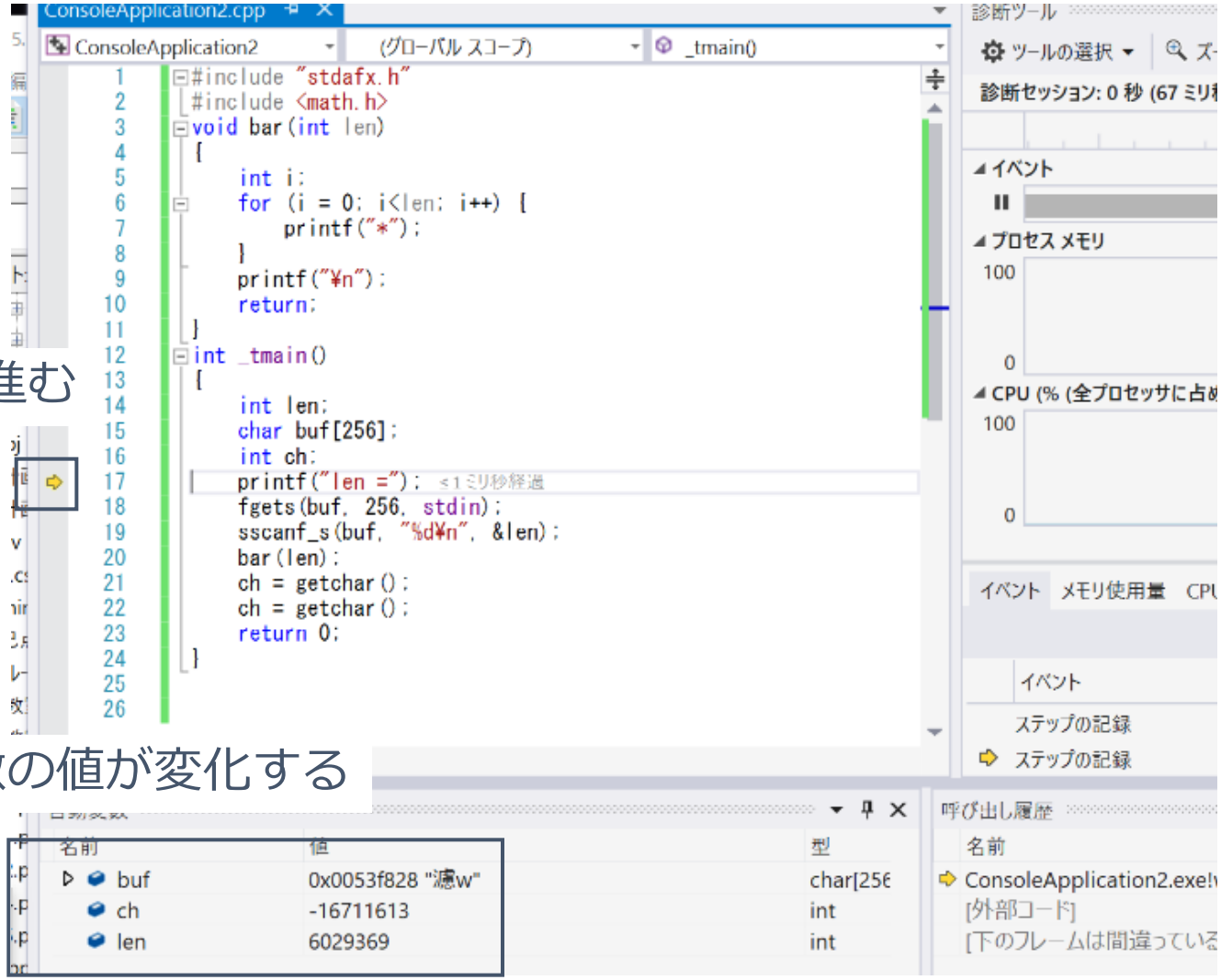
ビルドが正常終了したことを示すメッセージ

ステップ実行



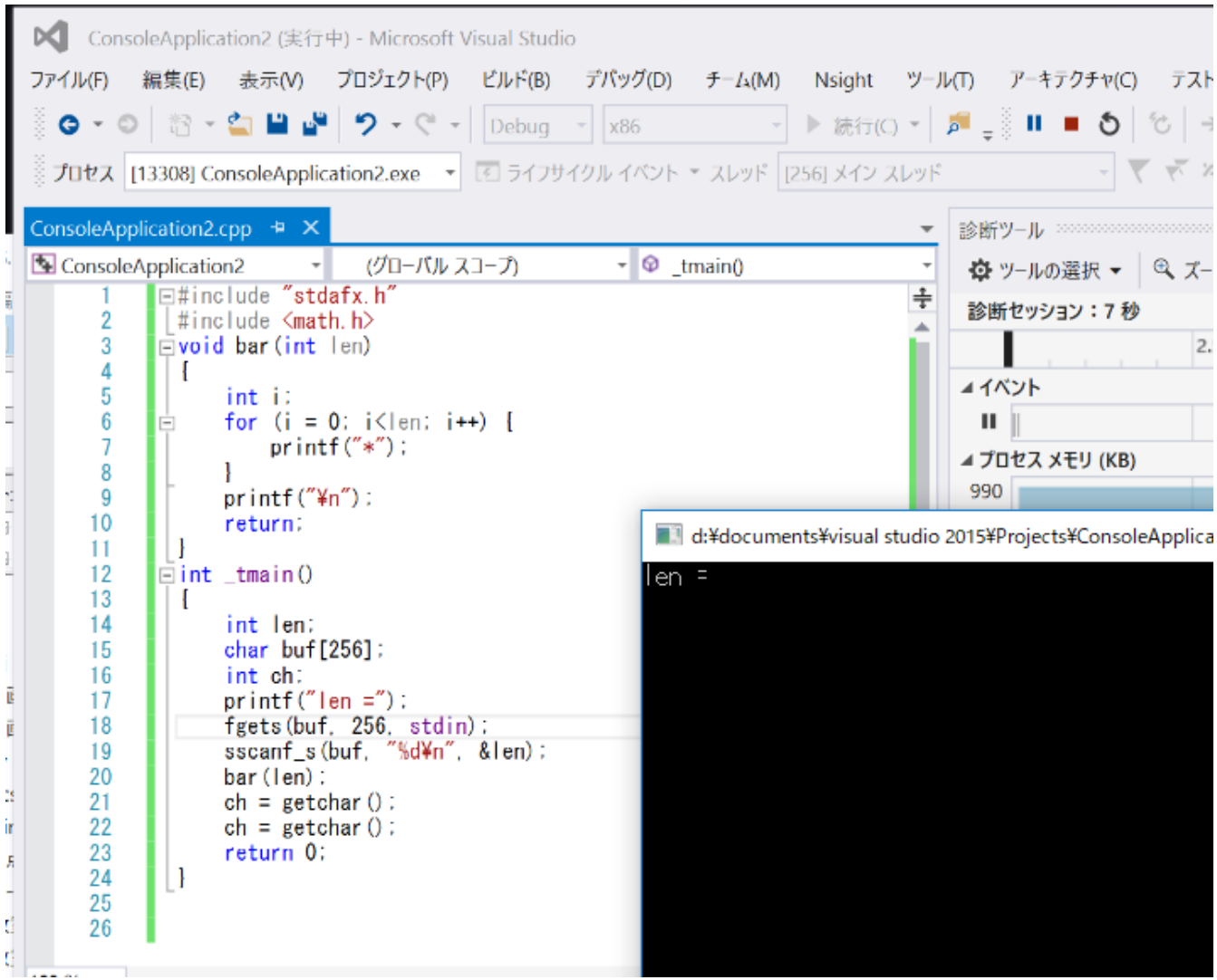
「**F10**」キーを押すとステップ実行が始まる．（マウスカーソルは，Microsoft Visual Studio C++ 内に入れておく）

マークが進む



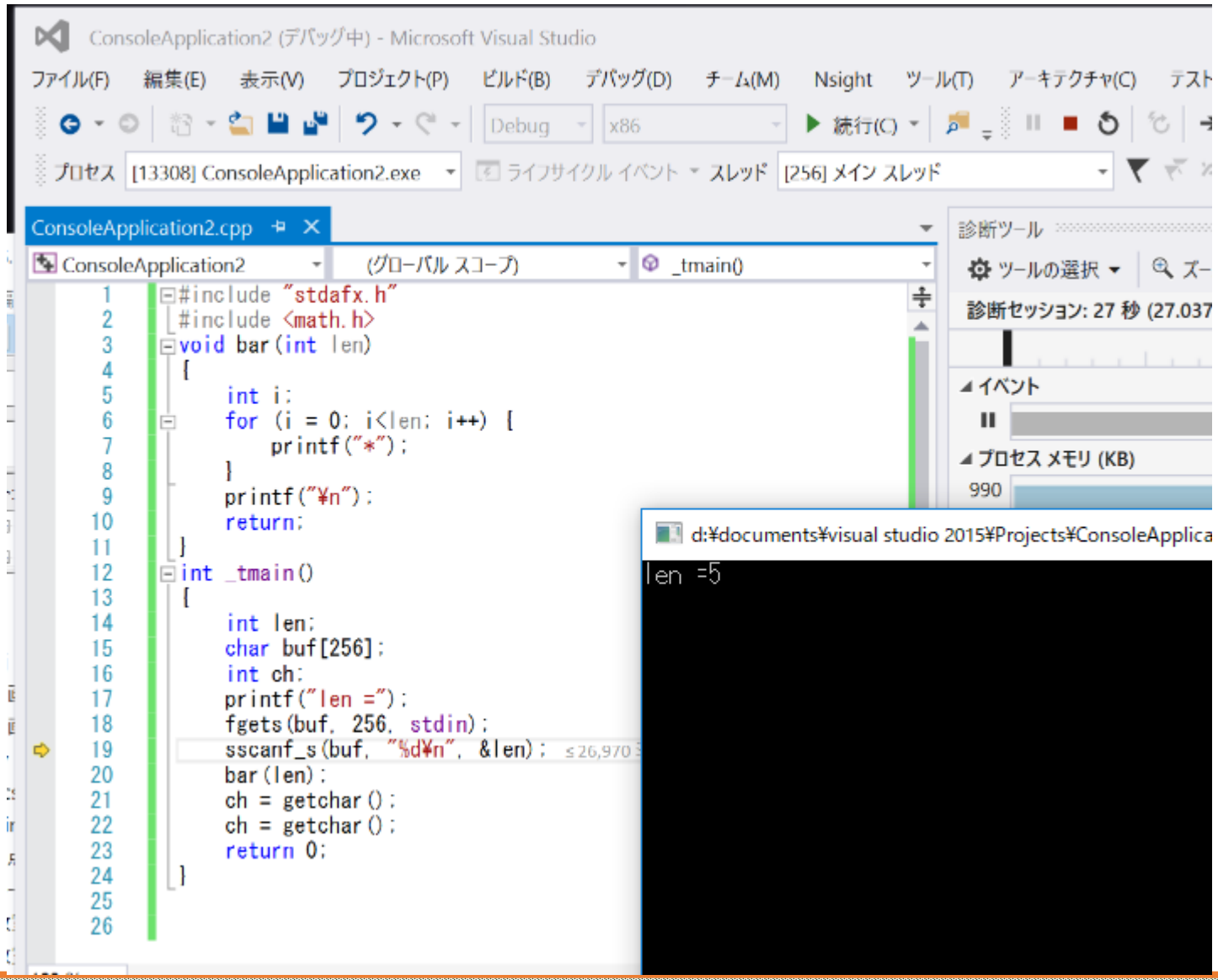
変数の値が変化する

さらに「**F10**」キーを押すとステップ実行が続く

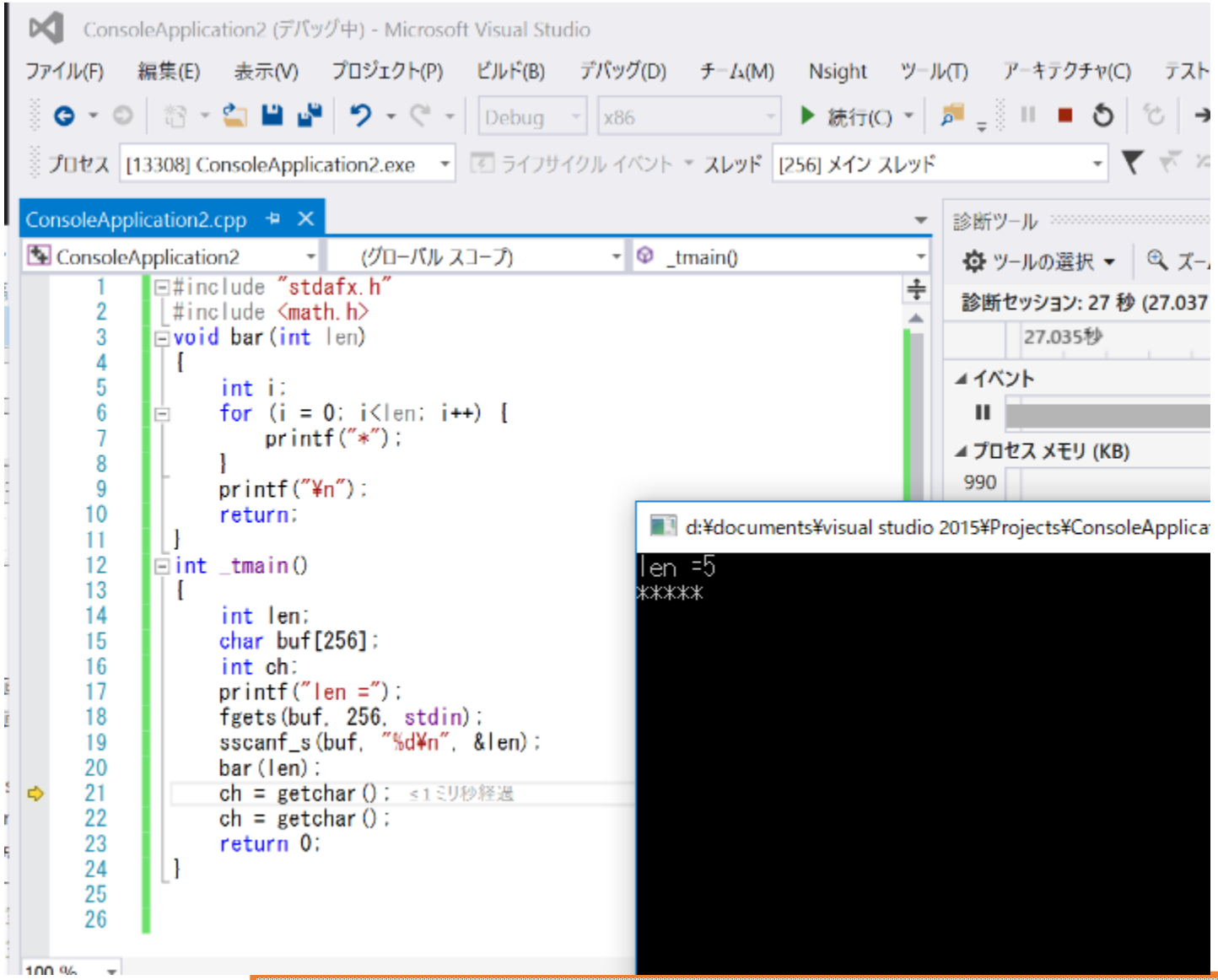



```
1  #include "stdafx.h"
2  #include <math.h>
3  void bar(int len)
4  {
5      int i;
6      for (i = 0; i<len; i++) {
7          printf("*");
8      }
9      printf("¥n");
10     return;
11 }
12 int _tmain()
13 {
14     int len;
15     char buf[256];
16     int ch;
17     printf("len =");
18     fgets(buf,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

さらに「**F10**」キーを押すとステップ実行が続く



実行ウインドウで「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）



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

```
#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;
}
```

プログラムの実行開始

メイン関数

関数呼び出し

bar(len);

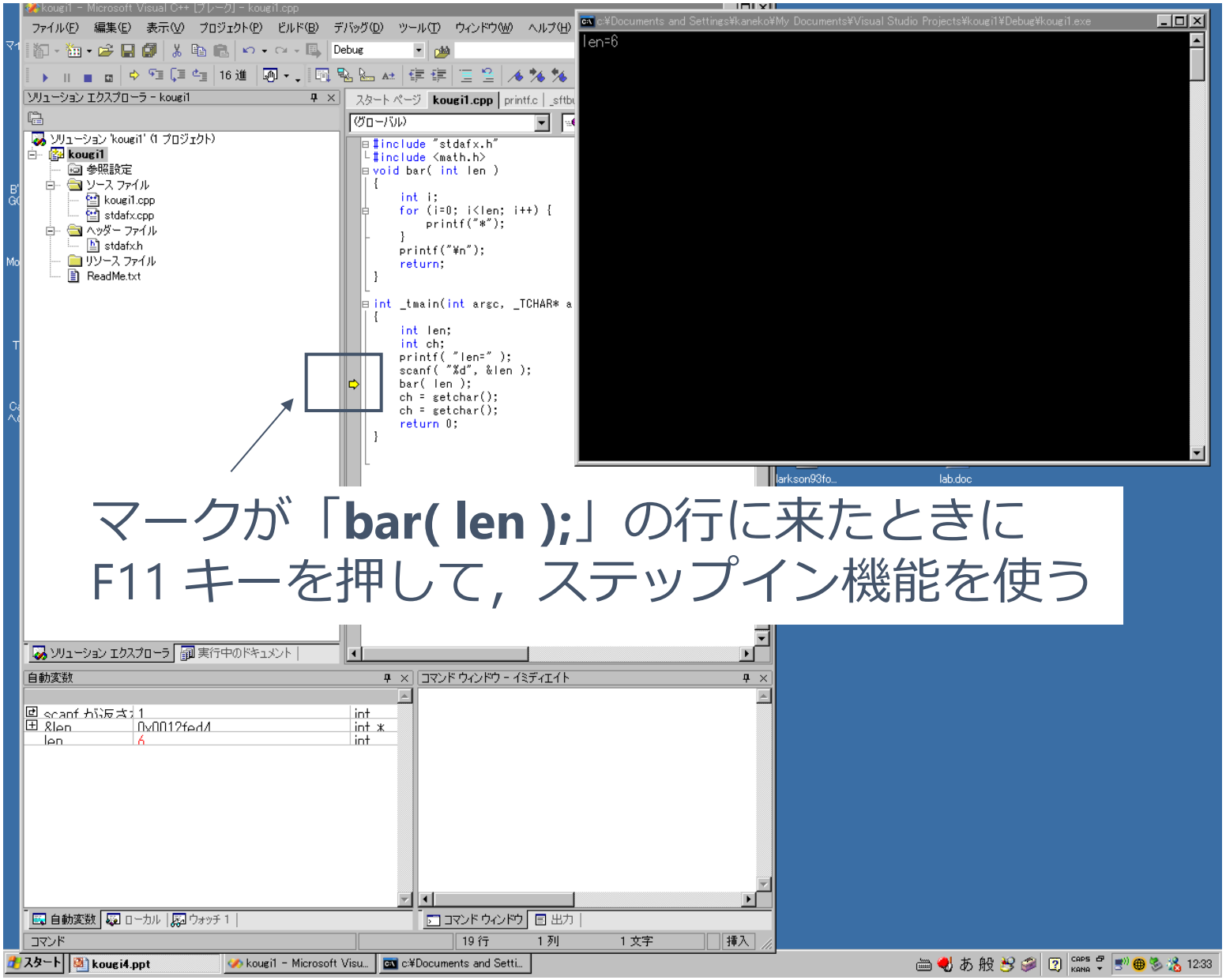
bar 関数

return;

プログラムの実行終了

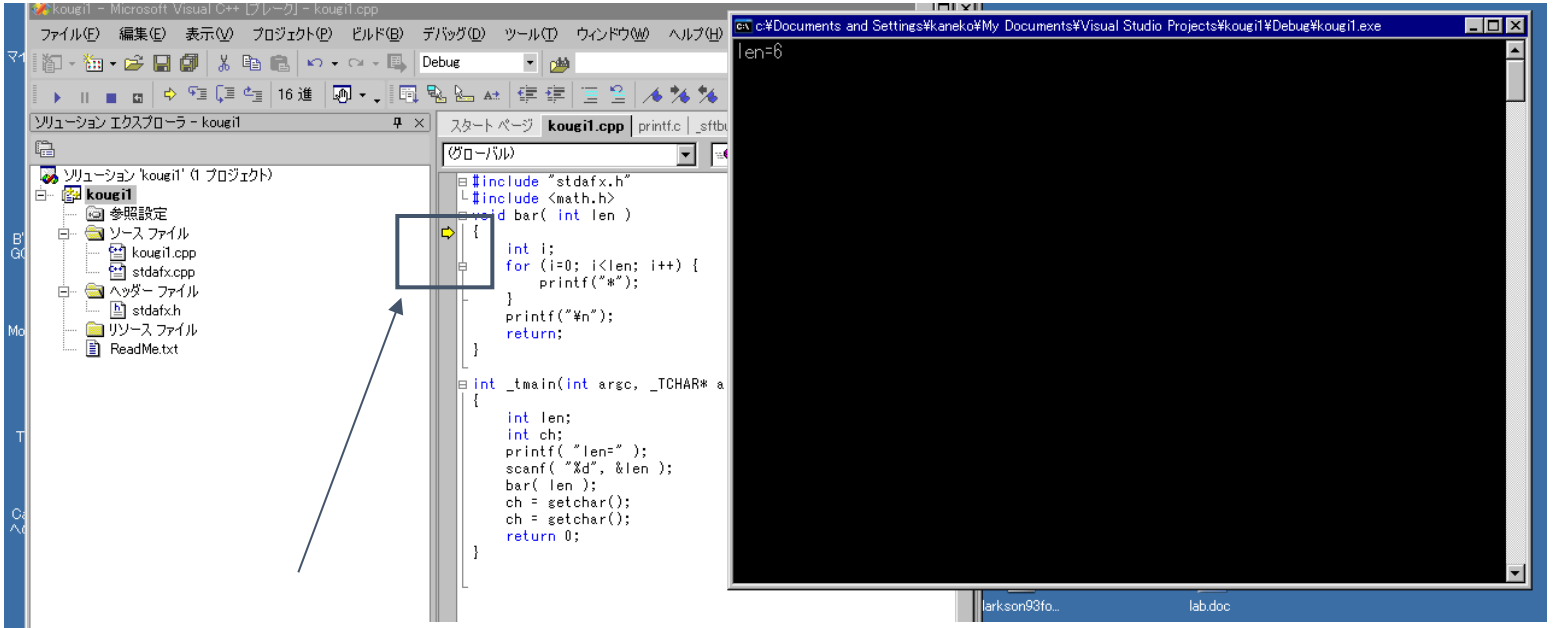
* printf, scanf の
呼び出しについては
図では省略

ステップイン機能

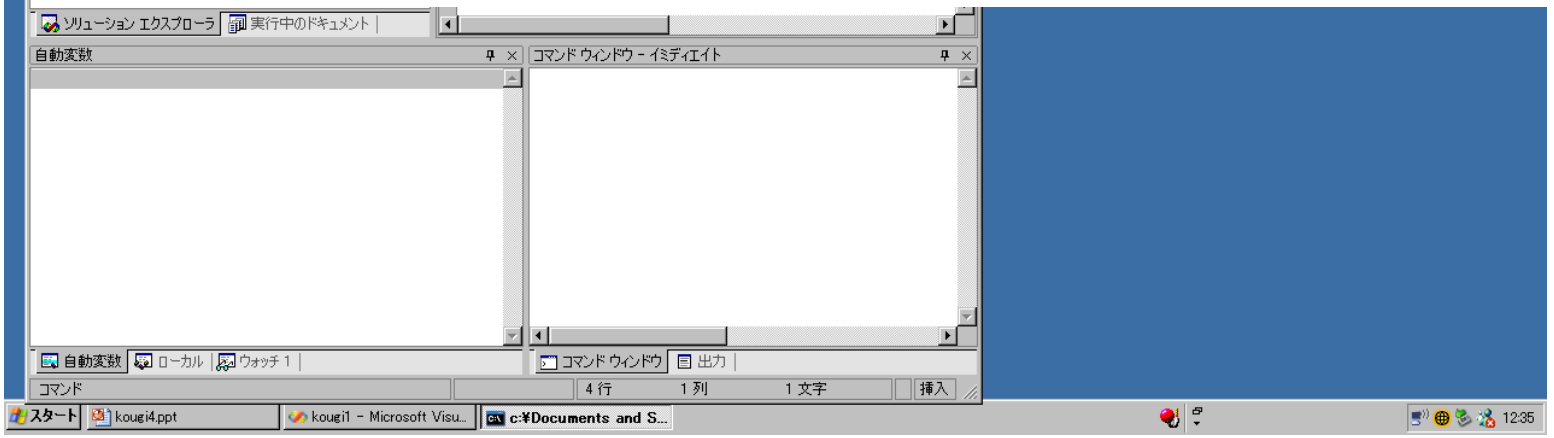


マークが「**bar(len);**」の行に来たときに
F11 キーを押して，ステップイン機能を使う

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）

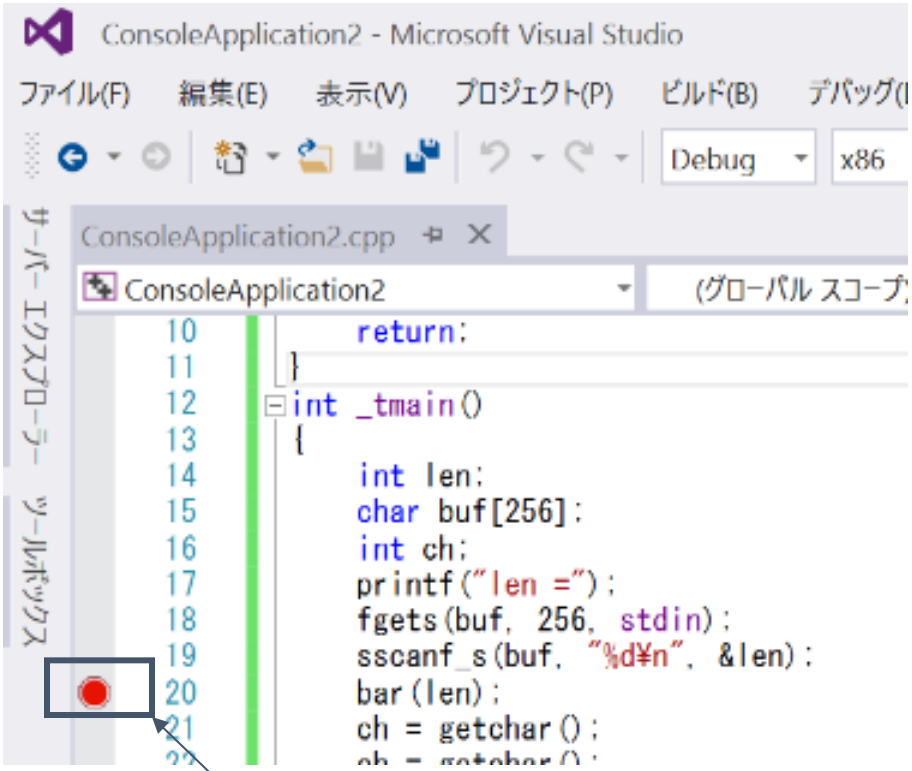


補足説明事項

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



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