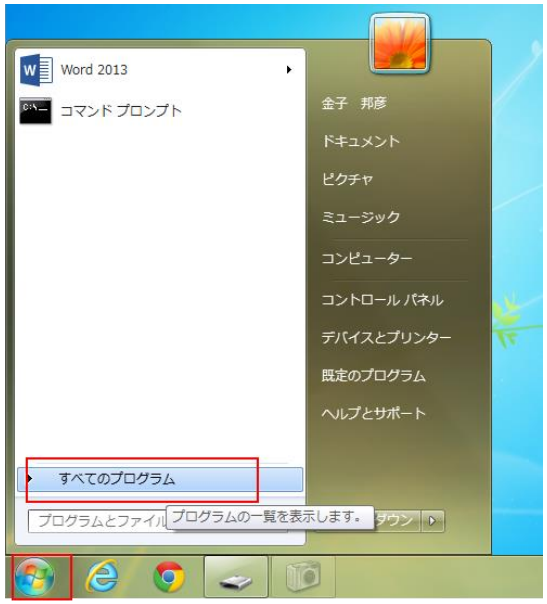
Visual Studio 2013 の起動とプロジェクトの新規作成

（C プログラミング入門）

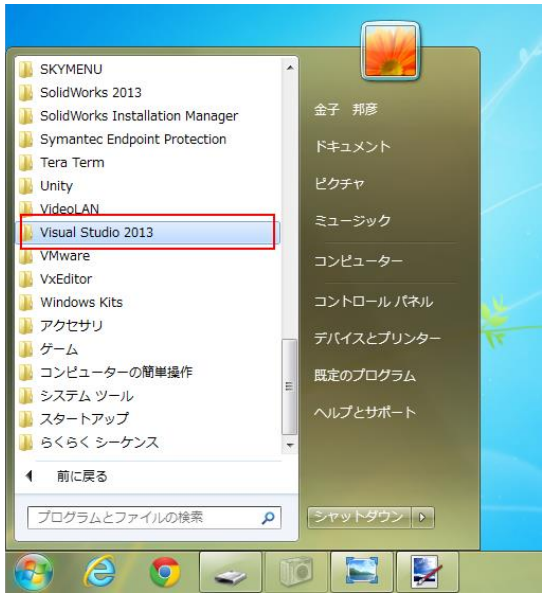
URL: https://www.kkaneko.jp/pro/adp/index.html

金子邦彦

Visual Studio 2013 の起動手順（例）



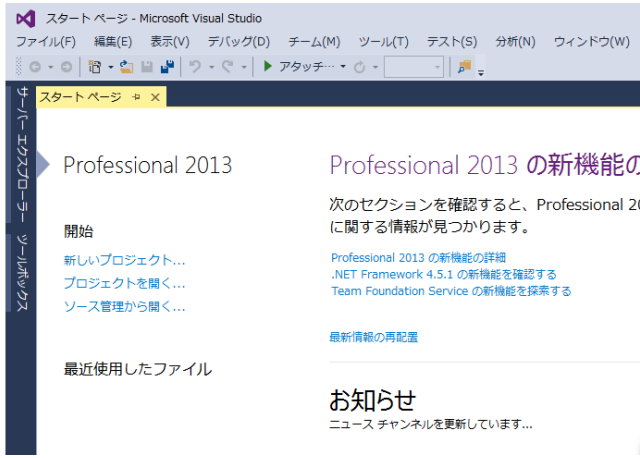
① 「スタートボタン」
→ 「すべてのプログラム」



② 「Visual Studio 2013」を展開

③ 「Visual Studio 2013」を選ぶ

Visual Studio 2013 の初回起動設定（初回起動時のみ）



① 「後で行う。」を選んでおく

② 「Visual Studio の開始」をクリック

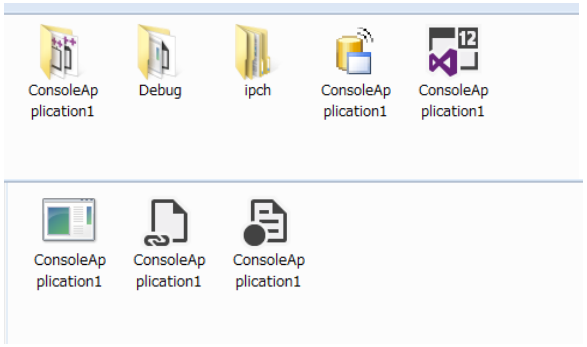
③ Visual Studio 2013 の画面が開く

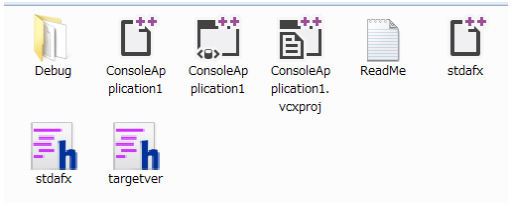
Visual Studio のプロジェクト

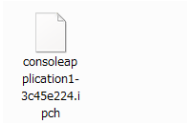
１つのソフトウエアに関するたくさんのファイルの集まり

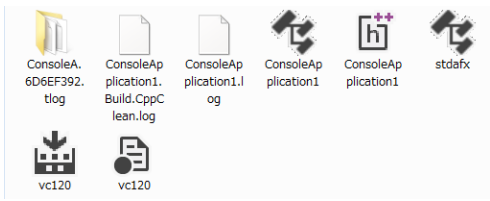
- プログラムが格納されたファイル（ソースファイル）
- 設定ファイル

など





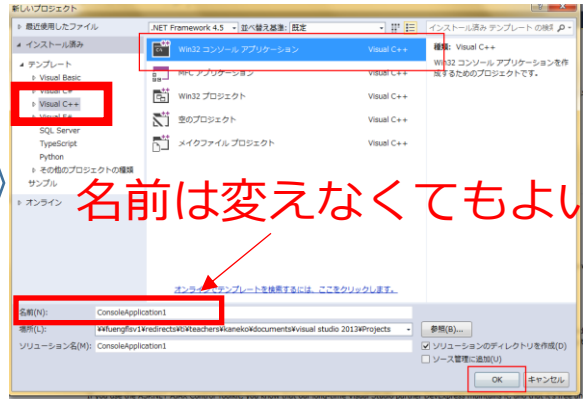




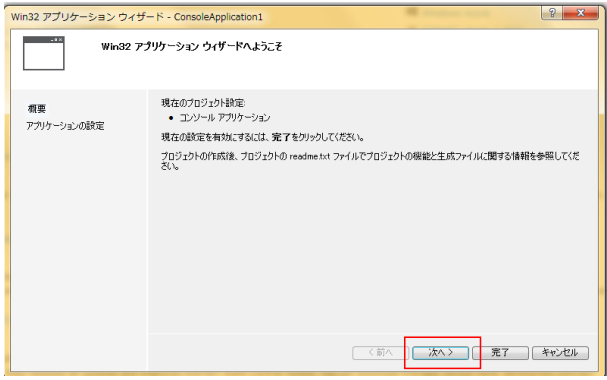
↑ある Visual Studio のプロジェクトのファイルとフォルダ

Visual Studio 2013 で Win 32 コンソールアプリケーション用 プロジェクトの新規作成 (1/2)

① 「ファイル」→「新規作成」
→「プロジェクト」



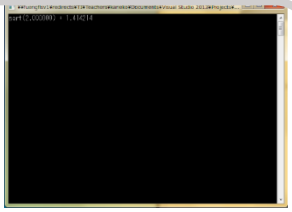
② 「Visual C++」→「Win32コンソールアプリケーション」→「OK」



③ 「次へ」

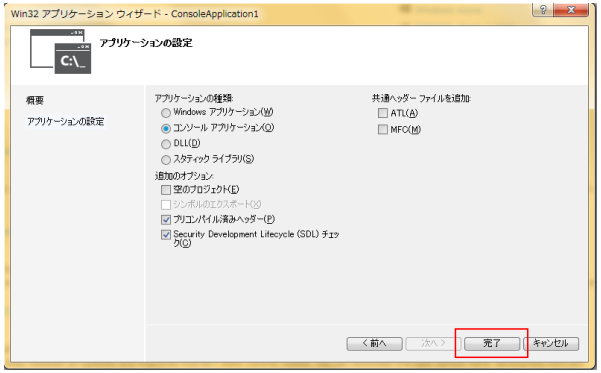
※ 次ページに続く

Win32コンソールアプリケーションとは，起動すると，Win32コンソール（例えば右図）が開くアプリのこと

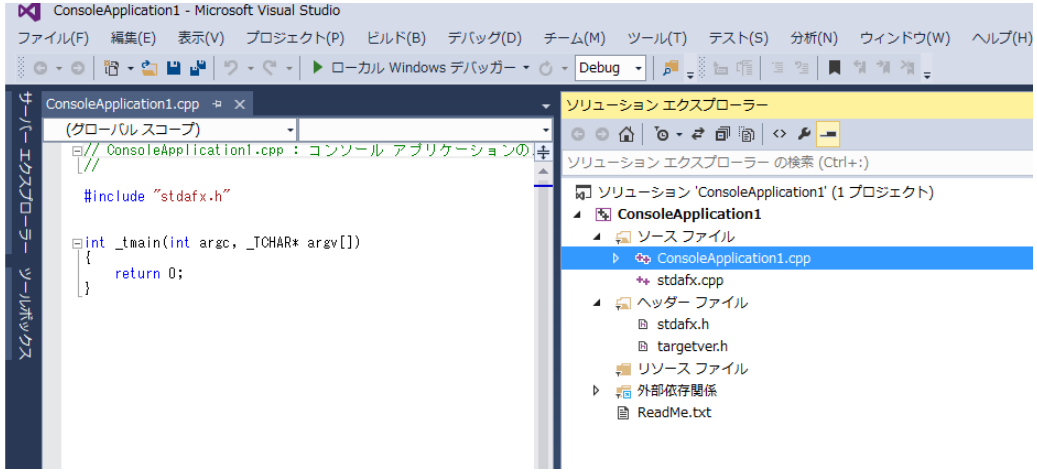


C++とは、C言語を機能アップしたプログラミング言語のこと

Visual Studio 2013 で Win 32 コンソールアプリケーション用 プロジェクトの新規作成 (2/2)

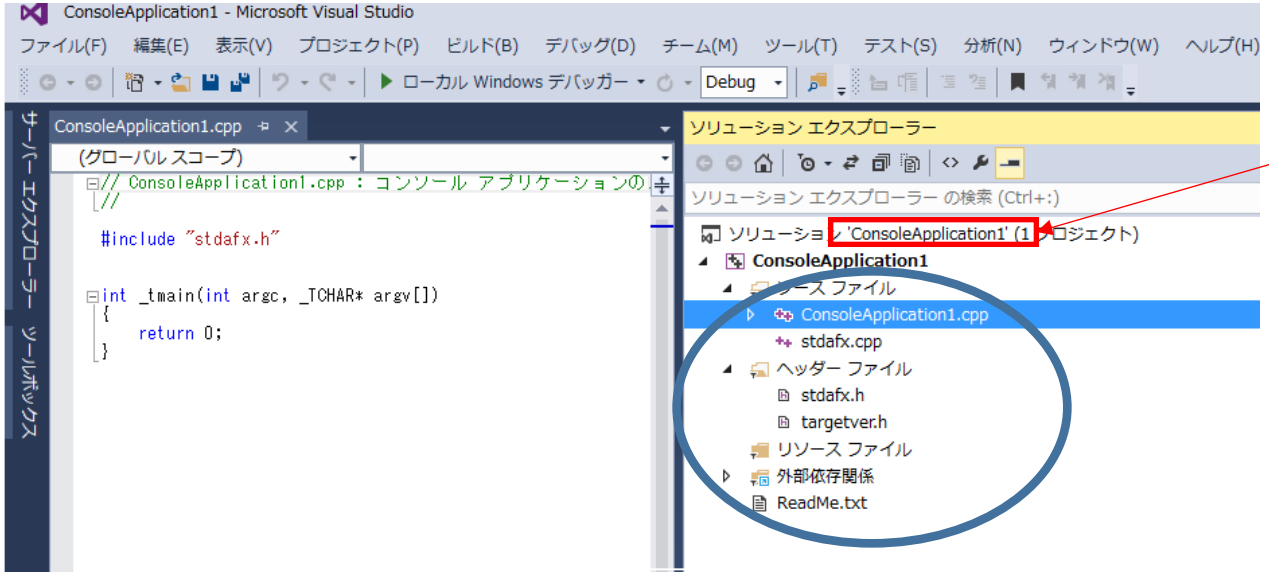


④ 「完了」をクリック



⑤ プロジェクトが新規作成されるので確認

Visual Studio 2013 の Win 32 コンソールアプリケーション用プロジェクト

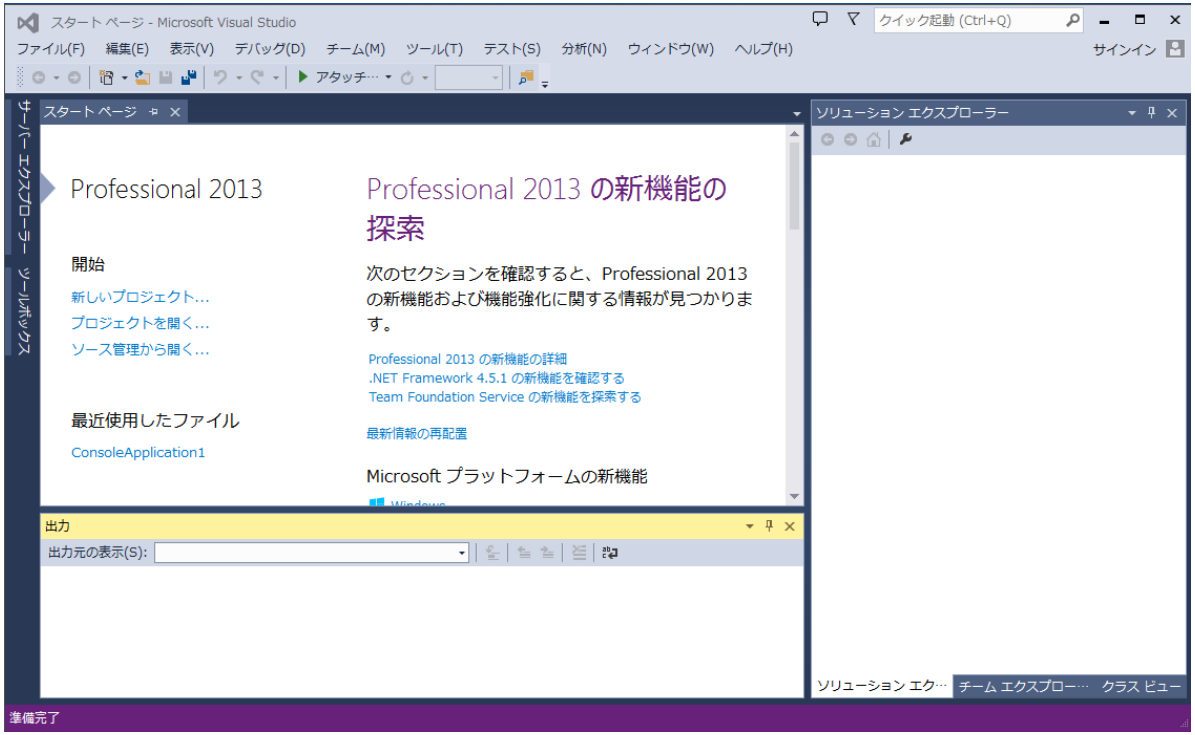


プロジェクトの名前

プロジェクトを構成するたくさんのファイル

※　表示されていないだけで、本当は、
もっとたくさんのファイルがある

Visual Studio 2013 を終了して、もう1度起動すると、画面がもとに戻る。

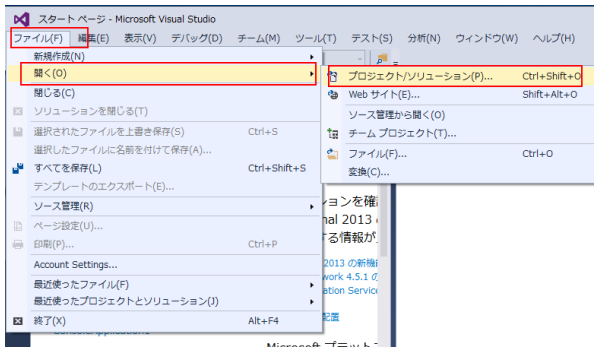


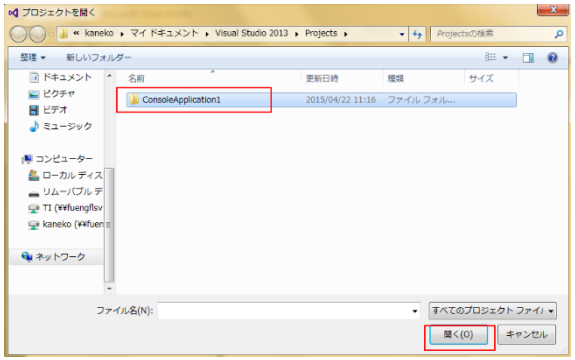
↑ Visual Studio 2013 を終了して、もう1度起動したところ

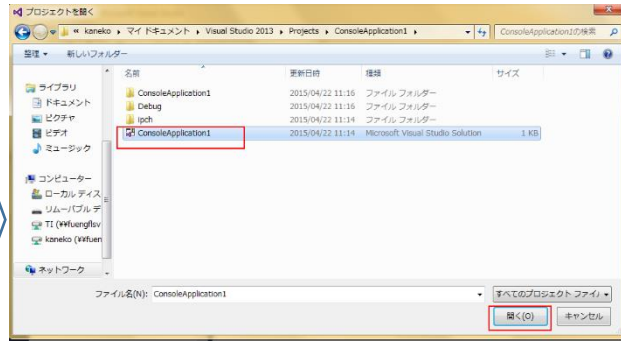
Visual Studio 2013 を終了して、もう1度起動すると、画面がもとに戻る。

⇒　プロジェクトの中身は消えていない

次の手順で作成済みのプロジェクトを開くことができる！







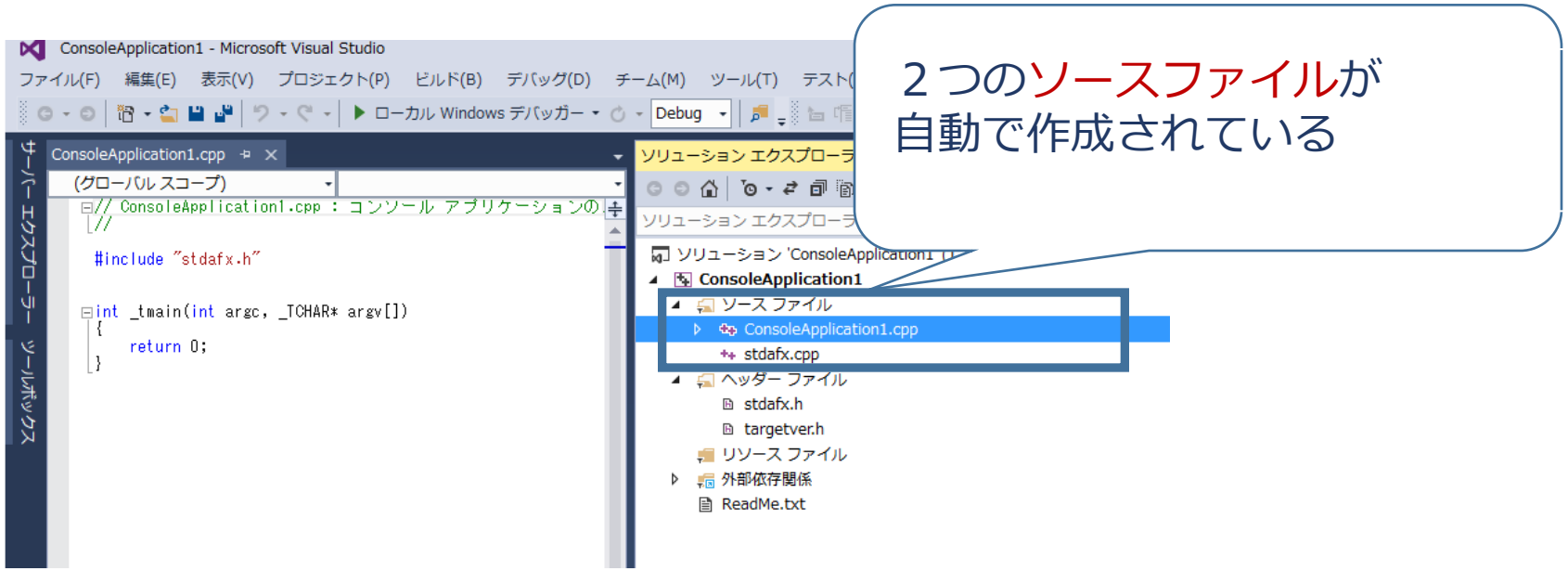
① 「ファイル」→「開く」→「プロジェクト／ソリューション」

②プロジェクト名と同じフォルダ名のフォルダを開く

③ Microsoft Visual Studio Solution ファイルを選び、「開く」をクリック

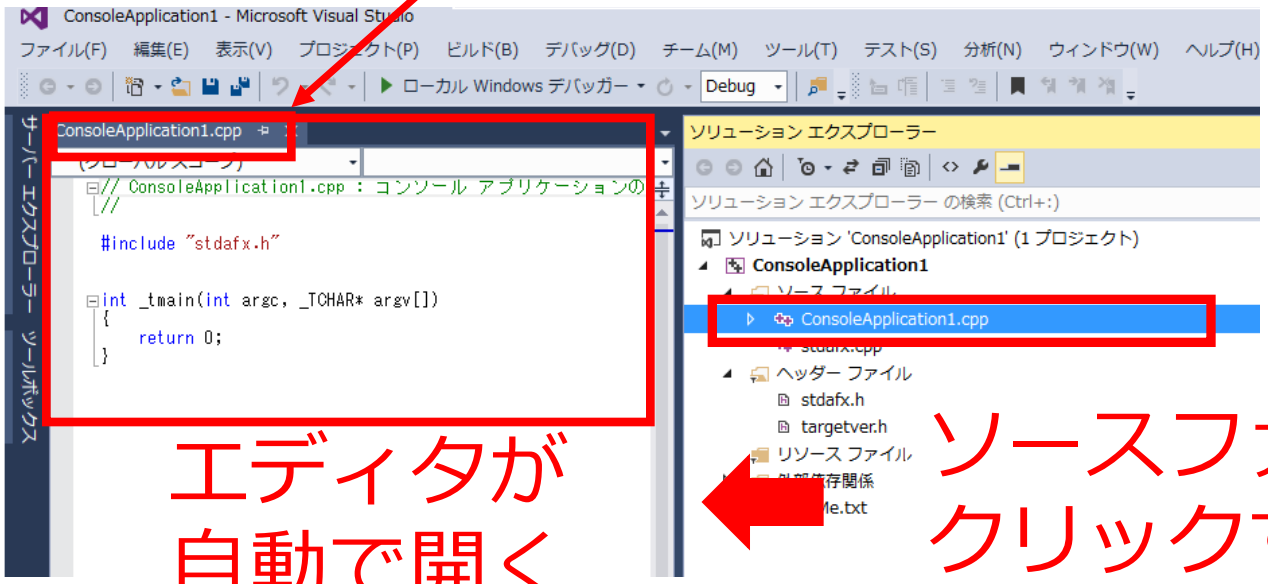
Visual Studio 2013 でビルドと実行

Visual Studio 2013 の Win 32 コンソールアプリケーション用プロジェクト



Visual Studio 2013 の Win 32 コンソールアプリケーション用プロジェクト

編集中のファイル名が表示



エディタが自動で開く

ソースファイルをクリックすると・・・

※ Win32 コンソールアプリケーション用プロジェクトを新規作成したとき、**エディタが自動で開く**ので、**そのまま使ってよい**

Visual C++ のソースファイル例

```
// ConsoleApplication1.cpp : コンソール アプリケーションのエントリ ポイン
//

#include "stdafx.h"
#include <math.h>

int _tmain(int argc, _TCHAR* argv[])
{
    double d = 2.0;
    printf("sqrt(%f) = %f¥n", d, sqrt(d));
    return 0;
}
```

← 1行追加

← ２行追加

自動生成されたプログラムを活用
（必要な分を追加）

Visual C++ のソースファイル例

```
// ConsoleApplication1.cpp : コンソール アプリケーションのエントリ ポイン
//

#include "stdafx.h"
#include <math.h>

int _tmain(int argc, _TCHAR* ar
{
    double d = 2.0;
    printf("sqrt(%f) = %f¥n", d, sqrt(d));
    return 0;
}
```

変数 d に、値 2.0 をセット

変数 d の平方根の計算。結果の表示

※ 「d = 2」と書かずに「d = 2.0」と書く理由は、小数付きの数として計算したい（整数ではない）ときの C 言語からの習慣

（参考）Linux での C 言語と Visual Studio の C++ 言語の違い

```
#include "stdafx.h"
#include <math.h>

int _tmain(int argc, _TCHAR* argv[])
{
    double d = 2.0;
    printf("sqrt(%f) = %f¥n", d, sqrt(d));
    return 0;
}
```

CentOS での C 言語	Visual Studio の C++ 言語
#include <stdio.h>	#include “stdafx.h”
int main ・・・	int _tmain(int argc, _TCHAR* argv[])

なぜ？ Windows 固有の機能も使いたいが、なるべく C 言語との違いを少なくしたいという工夫

プログラムに関する２種類のファイル

プログラムが格納されたファイル（ソースファイル）

その中身はプログラム

```
#include "stdafx.h"
#include <math.h>

int _tmain(int argc, _TCHAR* argv[])
{
    double d = 2.0;
    printf("sqrt(%f) = %f¥n", d, sqrt(d));
    return 0;
}
```

ソースファイルは、
テキストファイルの一種.
文字が格納されたファイルで、
各文字がコード化さている

実行型ファイル

その中身はマシン語（機械語）

```
00000000  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ..............
00000010  B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ........@.......
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00  00 00 00 00 F0 00 00 00  ................
00000040  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ........!..L.!Th
00000050  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS
00000070  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
00000080  FF FA 97 C6 BB 9B F9 95  BB 9B F9 95 BB 9B F9 95  ................
00000090  FD CA 24 95 B8 9B F9 95  FD CA 26 95 BA 9B F9 95  ..$.......&.....
000000a0  FD CA 19 95 A8 9B F9 95  FD CA 18 95 BC 9B F9 95  ................
000000b0  66 64 32 95 B9 9B F9 95  BB 9B F8 95 83 9B F9 95  fd2.............
000000c0  B6 C9 18 95 B9 9B F9 95  B6 C9 22 95 BA 9B F9 95  ..........".....
000000d0  B6 C9 27 95 BA 9B F9 95  52 69 63 68 BB 9B F9 95  ..'.....Rich....
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000f0  50 45 00 00 4C 01 07 00  A8 13 37 55 00 00 00 00  PE..L.....7U....
00000100  00 00 00 00 E0 00 02 01  0B 01 0C 00 00 3C 00 00  .............<..
```

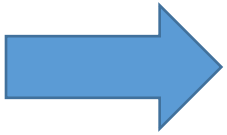
マシン語（機械語）とは
コンピュータに指令を与える
命令言語

ビルド（コンパイル）

ビルド（コンパイルともいう）とは、ソースファイルから、実行型ファイルを生成すること

```
#include "stdafx.h"
#include <math.h>

int _tmain(int argc, _TCHAR* argv[])
{
    double d = 2.0;
    printf("sqrt(%f) = %f¥n", d, sqrt(d));
    return 0;
}
```



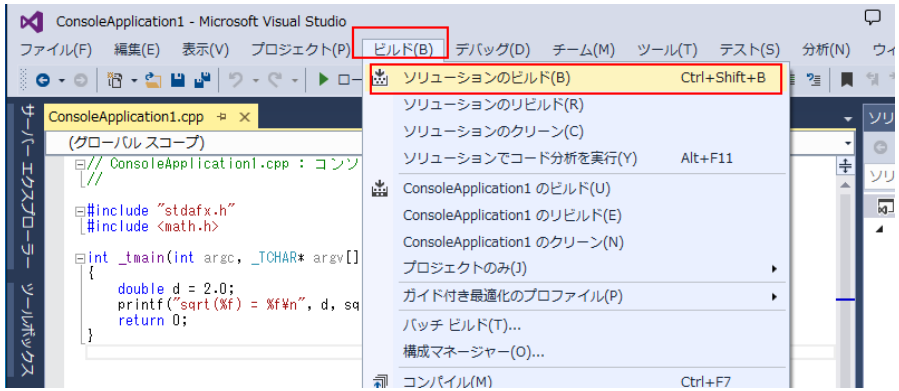
```
00000000  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ..............
00000010  B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ........@.......
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00  00 00 00 00 F0 00 00 00  ................
00000040  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ........!..L.!Th
00000050  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS
00000070  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
00000080  FF FA 97 C6 BB 9B F9 95  BB 9B F9 95 BB 9B F9 95  ................
00000090  FD CA 24 95 B8 9B F9 95  FD CA 26 95 BA 9B F9 95  ..$.......&.....
000000a0  FD CA 19 95 A8 9B F9 95  FD CA 18 95 BC 9B F9 95  ................
000000b0  66 64 32 95 B9 9B F9 95  BB 9B F8 95 83 9B F9 95  fd2.............
000000c0  B6 C9 18 95 B9 9B F9 95  B6 C9 22 95 BA 9B F9 95  ..........".....
000000d0  B6 C9 27 95 BA 9B F9 95  52 69 63 68 BB 9B F9 95  ..'.....Rich....
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000f0  50 45 00 00 4C 01 07 00  A8 13 37 55 00 00 00 00  PE..L.....7U....
00000100  00 00 00 00 E0 00 02 01  0B 01 0C 00 00 3C 00 00  .............<..
```

ビルド
（コンパイルともいう）

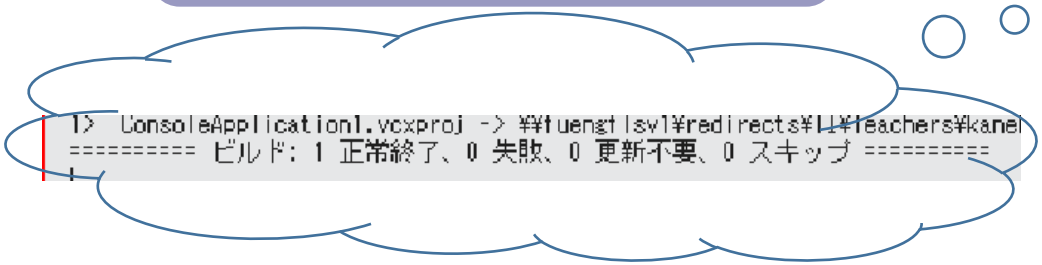
ソースファイル

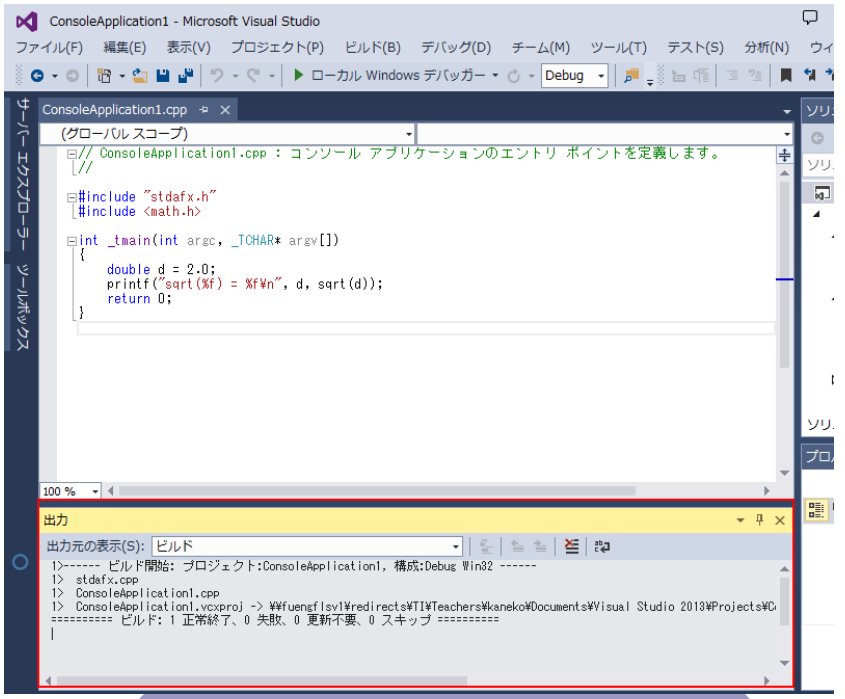
実行型ファイル

Visual Studio 2013 でのビルド手順



① 「ビルド」
→「ソリューションのビルド」

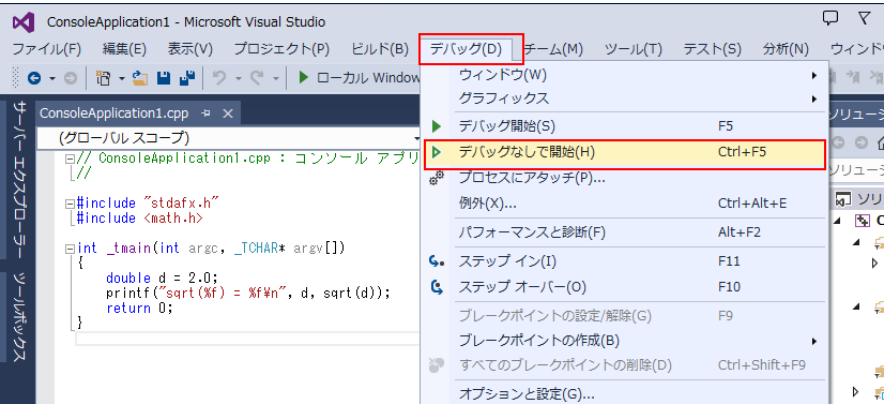




② 必ず「1　正常終了、0　失敗」の表示を確認

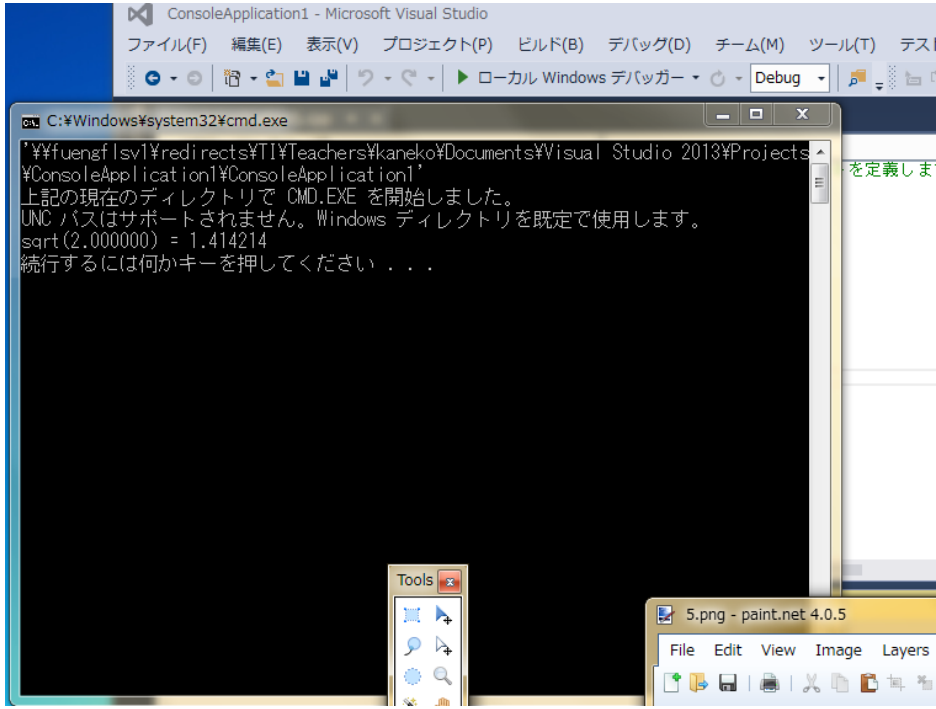
ビルド：1　正常終了、0　失敗

Visual Studio 2013 での実行手順



① 「デバッグ」
→ 「デバッグなしで開始」

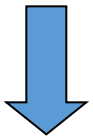
ビルドが正常終了したら、
実行できる



② Win32 コンソールが開く

プログラムのビルドと実行

ソースファイル



実行型ファイル

ビルド
(build)

実行
(execute)