vc-2. Visual Studio C++ のデバッガー

（Visual Studio C++ の機能と操作演習，全5回）

https://www.kkaneko.jp/cc/vc/index.html

金子邦彦

目次

2-1 デバッガーの機能

2-1 デバッガー

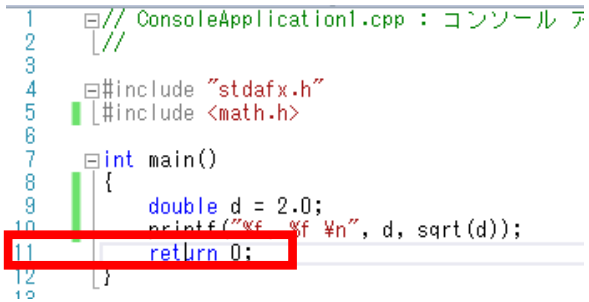
デバッガーとは、プログラムの不具合（バグ）の発見や修正を支援するソフトウェアのこと

※ **バグ**を自動的に取り除いてくれるわけではない

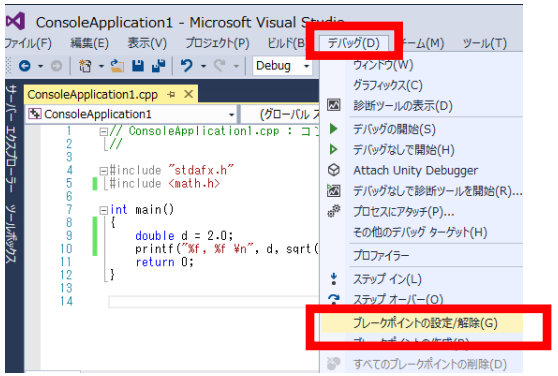
主な機能

- **ブレークポイント**機能：　プログラムの実行中断
- **トレース**機能：　プログラム実行中に**変数**の値などを表示

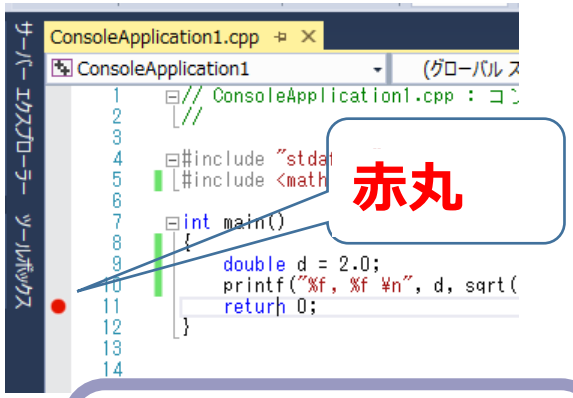
Visual Studio 2015 でのブレークポイント設定手順



① 「**return 0;**」の行をマウスでクリック



② 「デバッグ」→「**ブレークポイントの設定/解除**」

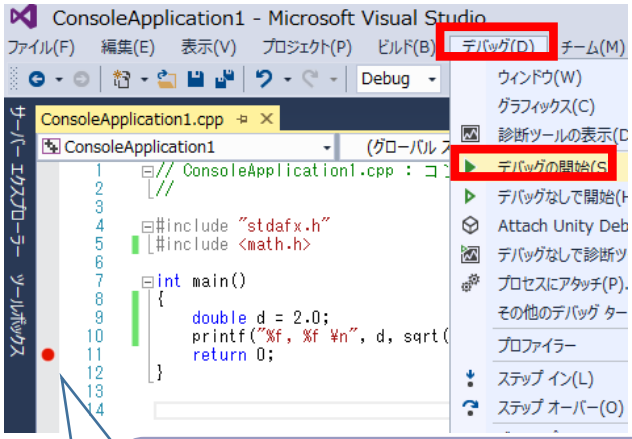


③ ブレークポイントが設定されるので**確認**

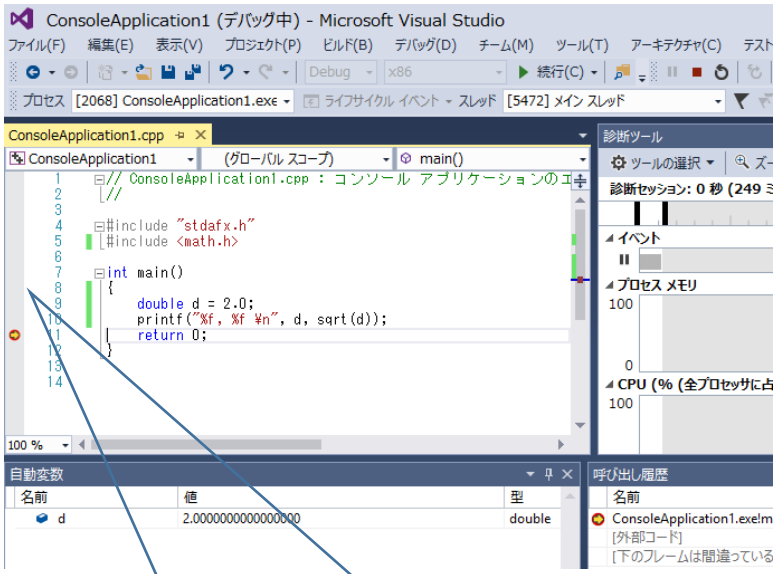
ブレークポイントとは、プログラムの実行を中断させたい行．複数設定可

ブレークポイントを解除したいときは赤丸をクリック
※ 同様の操作で、**ブレークポイント**の設定もできる

Visual Studio 2015 でのデバッガー起動手順



① 「デバッグ」
→ 「**デバッグの開始**」

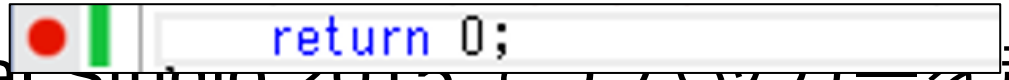


ブレークポイントが設定された状態で、**デバッガー**を起動しようとしている

形が変化！　これは、プログラムの実行が、ここで中断していることを示す

パソコン演習

- Visual Studio 2015 で「return 0;」の行にブレークポイントを設定しなさい



- Visual Studio 2015 でデバッガーを起動しなさい.「return 0;」の行で、実行が中断することを確認しなさい

- あとで使うので、中断したままにしておくこと

2-2 ステップ実行機能

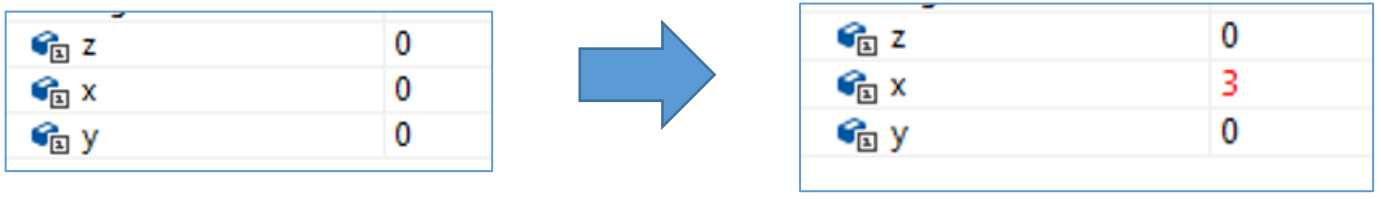
2-1 ステップ実行機能

ステップ実行機能を使うことで、

- 変数の値の変化

  ※変数名と値の対応表

- メモリ中のデータの変化　※ダンプリスト形式



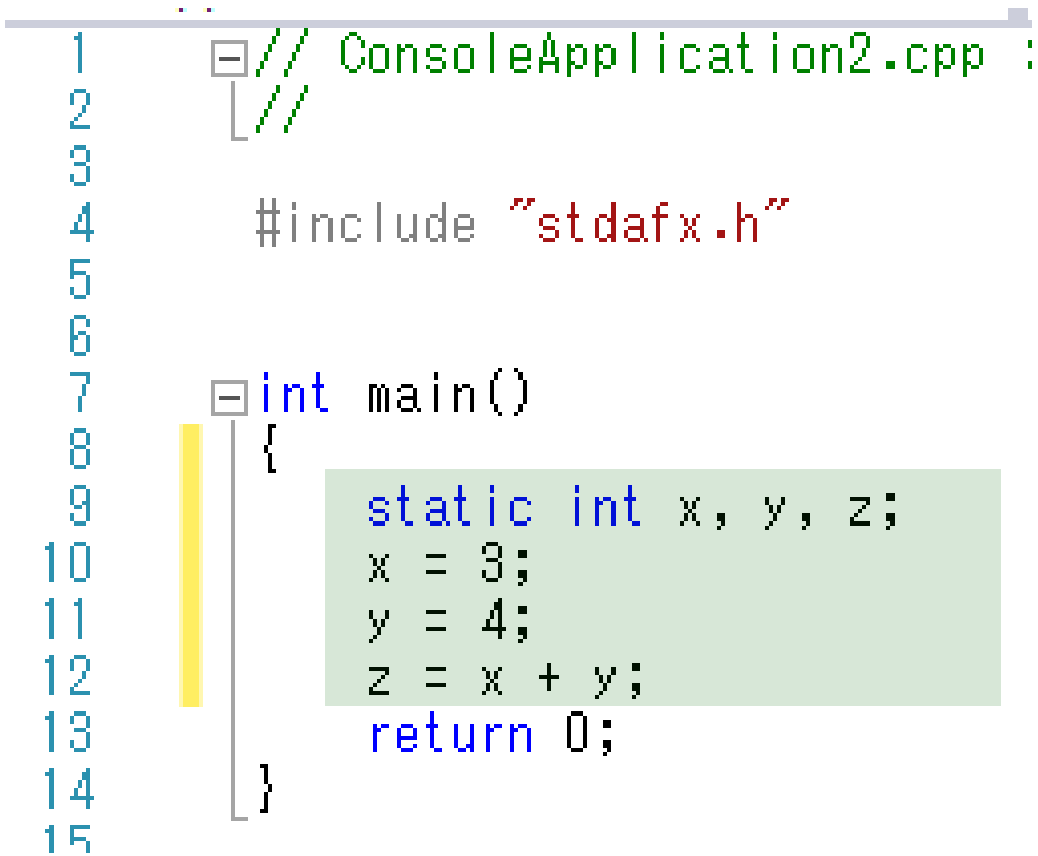
変数 x の値が 0 から 3 に変化
<ステップ実行機能で表示できる>

ステップ実行の種類

- ステップオーバー　F10キー
  - プログラム実行を１行進める．関数の中に入らない
- ステップイン　F11キー
  - プログラム実行を１行進める．関数の中に入る
- ステップアウト　Shift と F1１キー同時押し
  - 関数の外に出る

※ 「関数」は、
プログラミング言語の関数のこと

Visual C++ のソースファイル例



```
1   // ConsoleApplication2.cpp :
2   //
3
4   #include "stdafx.h"
5
6
7   int main()
8   {
9       static int x, y, z;
10      x = 3;
11      y = 4;
12      z = x + y;
13      return 0;
14  }
15
```

← ４行追加

自動生成されたプログラムを活用（４行追加するだけ）

Visual C++ のソースファイル例

```
1   // ConsoleApplication2.cpp :
2   //
3
4   #include "stdafx.h"
5
6
7   int main()
8   {
9       static int x, y, z;
10      x = 3;
11      y = 4;
12      z = x + y;
13      return 0;
14  }
15
```

変数 x に値 3 をセット
変数 y に値 4 をセット

x と y の値を足して、
結果を z に格納

※ 「static int x, y, z;」の「static」は、変数 x, y, z のメモリへの格納法をコントロールするキーワード

変数の値の変化

```
x = 3;
y = 4;
z = x + y;
```

この行で、 x の値は 3 に変化

この行で、 y の値は 4 に変化

この行で、 z の値は 7 に変化

パソコン演習

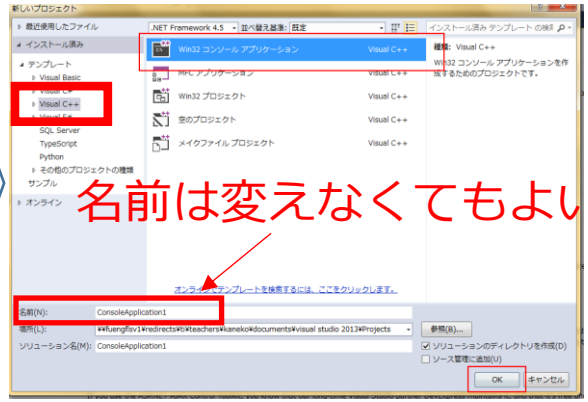
ステップオーバー機能

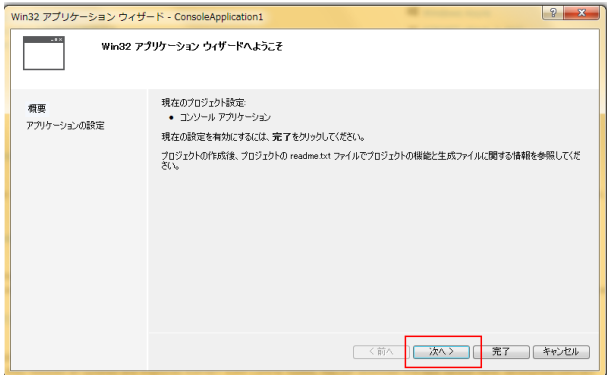
- Visual Studio 2015 を起動しなさい

- Visual Studio 2015 で、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なさい

プロジェクトの「名前」は何でもよい

Visual Studio 2015 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)





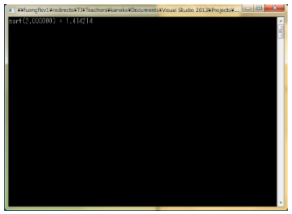
① 「ファイル」→「新規作成」→「**プロジェクト**」

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

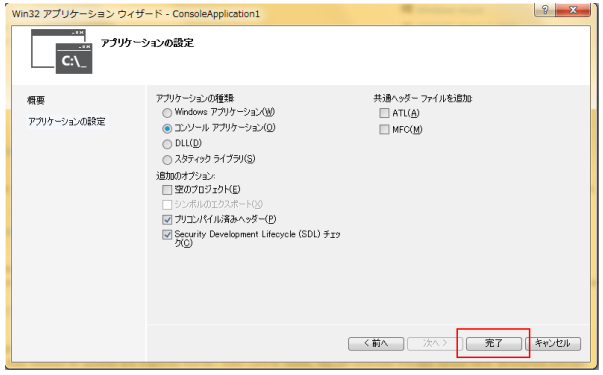
③ 「次へ」

※ 次ページに続く

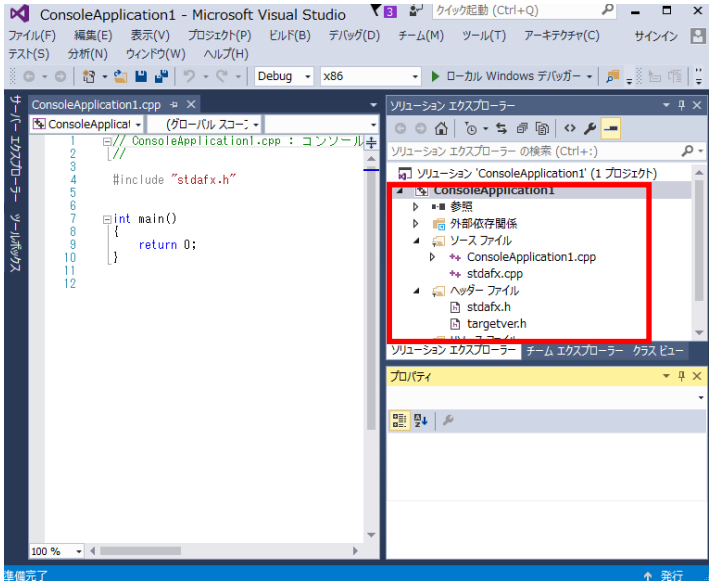
Win32コンソールアプリケーションとは，起動すると，**Win32コンソール**（例えば右図）が開くアプリのこと



Visual Studio 2015 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)

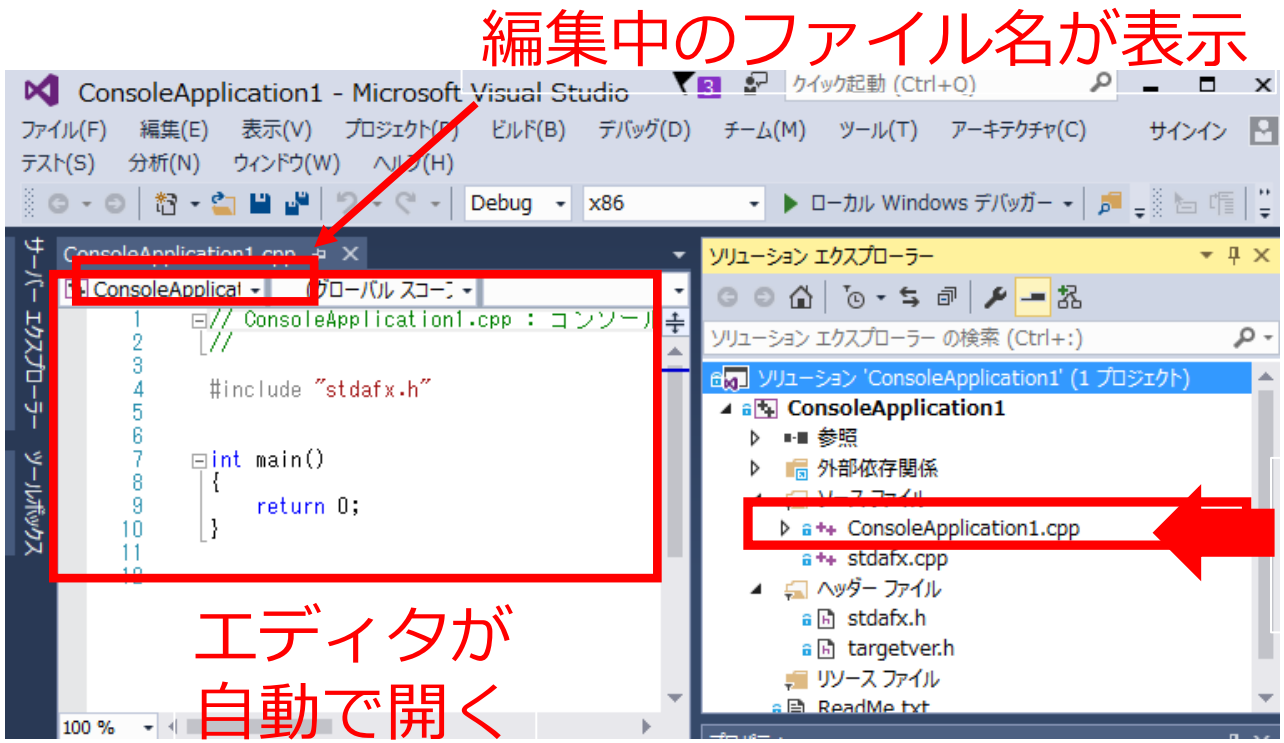


④「**完了**」をクリック



⑤ **プロジェクト**が新規作成されるので確認

Visual Studio 2015 の Win 32 コンソールアプリケーション用プロジェクト



※ 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**

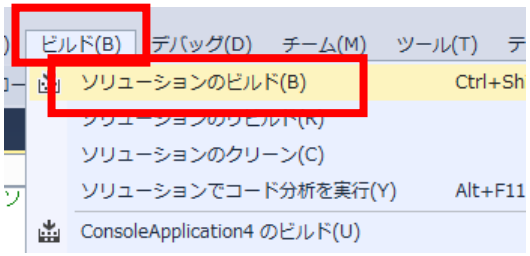
- Visual Studio 2015 のエディタを使って、ソースファイルを編集しなさい

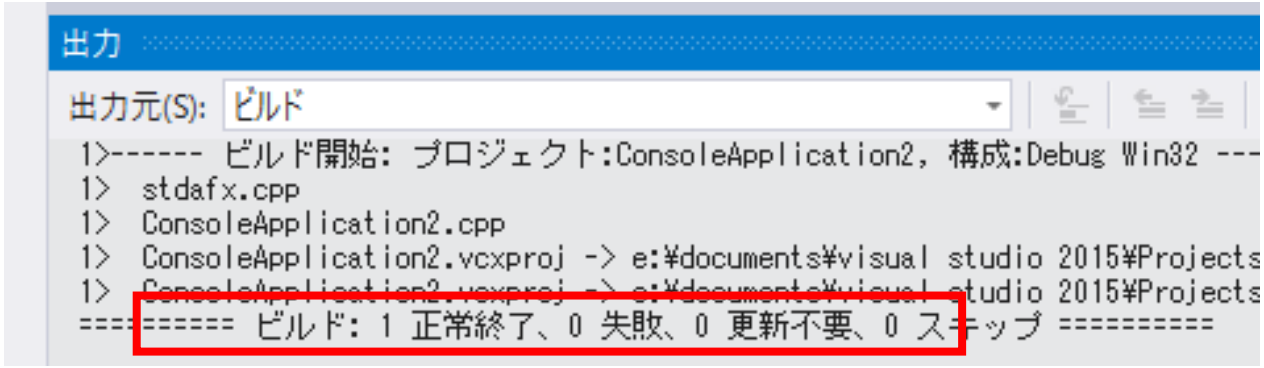
```
1   // ConsoleApplication2.cpp :
2   //
3
4   #include "stdafx.h"
5
6
7   int main()
8   {
9       static int x, y, z;
10      x = 3;
11      y = 4;
12      z = x + y;
13      return 0;
14  }
15
```

← **４行追加**

- ビルドしなさい．ビルドのあと「１　正常終了、０　失敗」の表示を確認しなさい

→　表示されなければ、プログラムのミスを自分で確認し、修正して、ビルドをやり直す



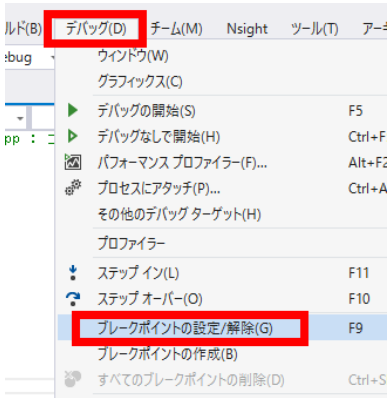


- Visual Studio 2015 で「x = 3;」の行に、ブレークポイントを設定しなさい

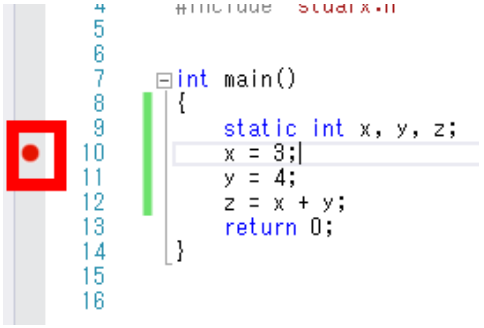
```
#include "stdafx.h"


int main()
{
    static int x, y, z;
    x = 3;
    y = 4;
    z = x + y;
    return 0;
}
```

① 「x = 3;」の行をマウスでクリック

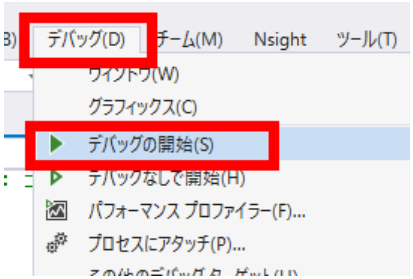


② 「デバッグ」→「**ブレークポイントの設定/解除**」



③ ブレークポイントが設定されるので確認
赤丸がブレークポイントの印

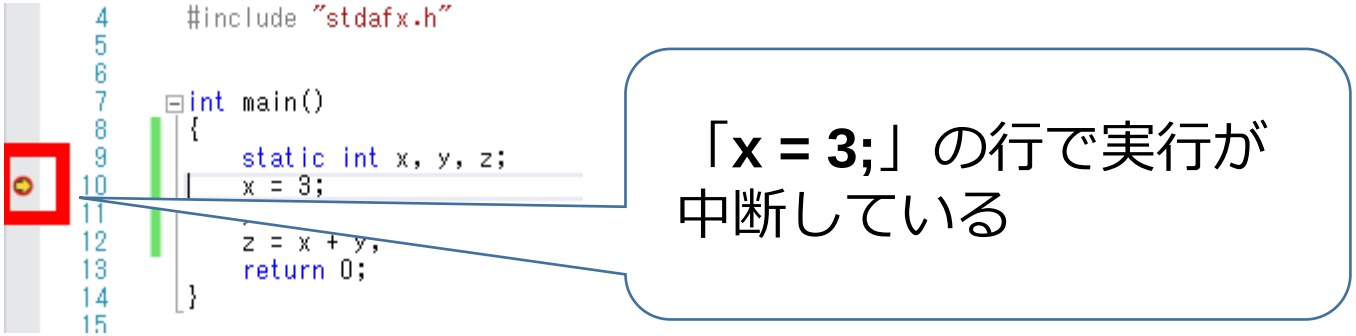
- Visual Studio 2015 で、デバッガーを起動しなさい.



「デバッグ」
→「**デバッグの開始**」

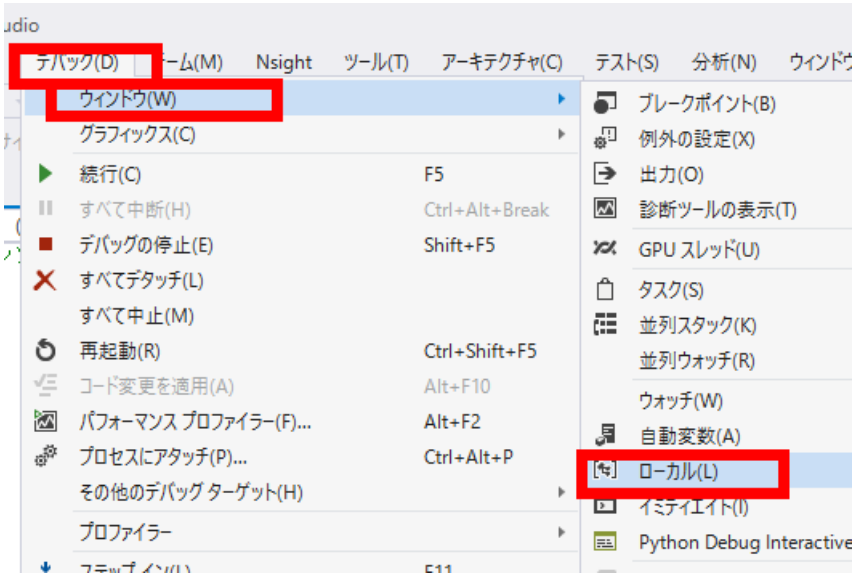
- 「x = 3;」の行で、実行が中断することを確認しなさい

  あとで使うので、中断したままにしておくこと

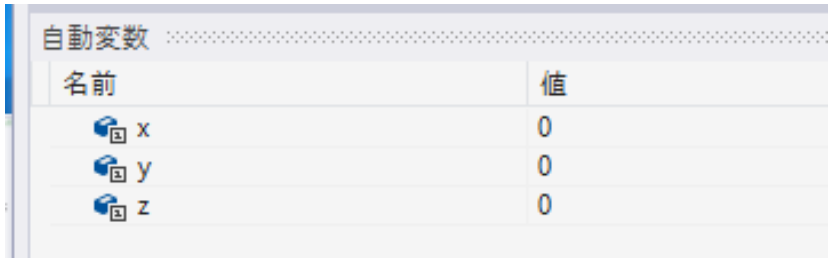


「**x = 3;**」の行で実行が中断している

- 「x = 3;」の行で、実行が中断した状態で、変数の値を表示させなさい．手順は次の通り．



①「デバッグ」
→「ウインドウ」
→ 「**ローカル**」

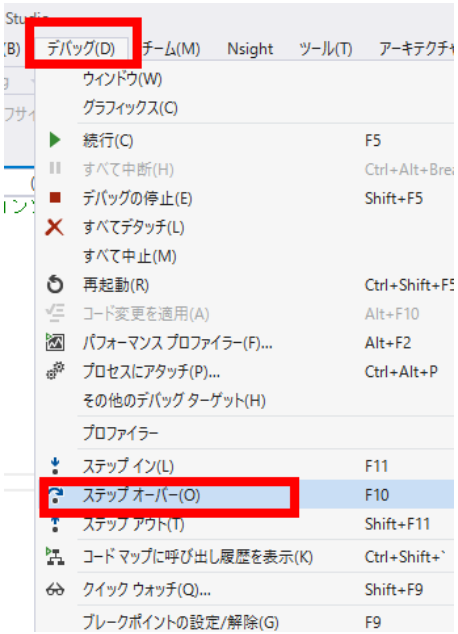


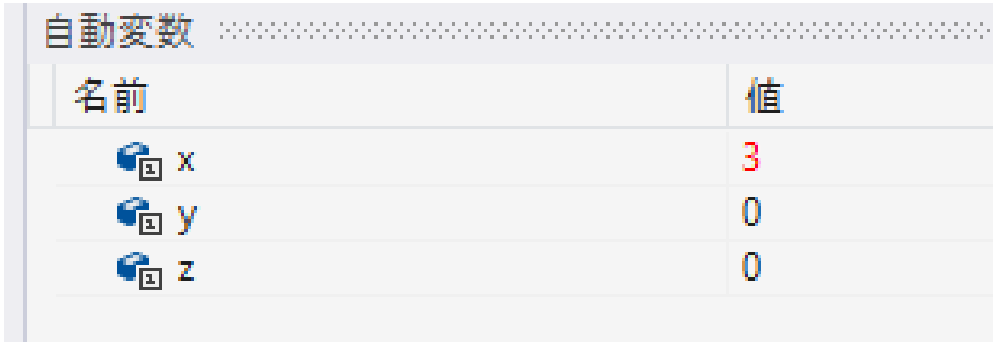
変数 x, y, z の値はすべて 0

② 変数名と値の対応表が表示される

「**x = 3;**」は未実行で　あることを確認！

- １回だけステップオーバーの操作を行い、変数 x の値が 3 に変化することを確認しなさい.



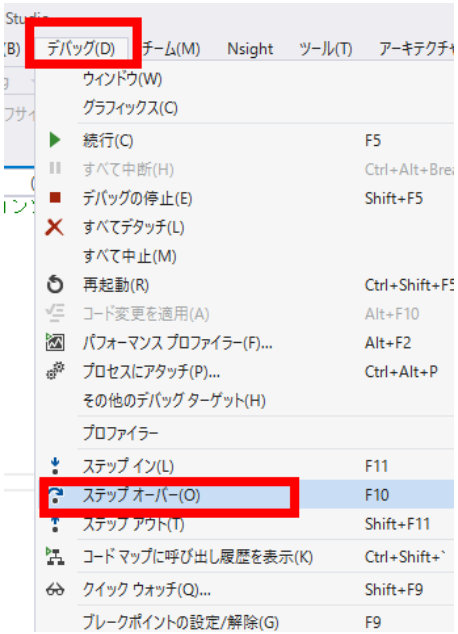


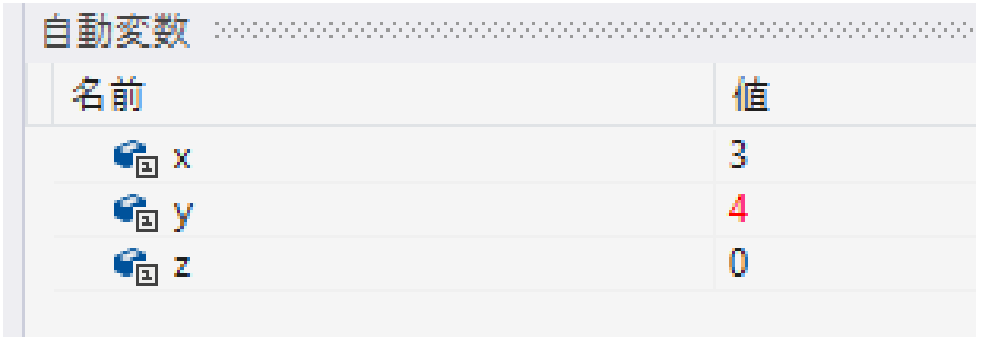
変数 x の値が 3 に変化！

「デバッグ」
→「**ステップオーバー**」
（あるいは **F10 キー**）

「**x = 3;**」が実行された

- もう１回だけステップオーバーの操作を行い、変数 y の値が 4 に変化することを確認しなさい.



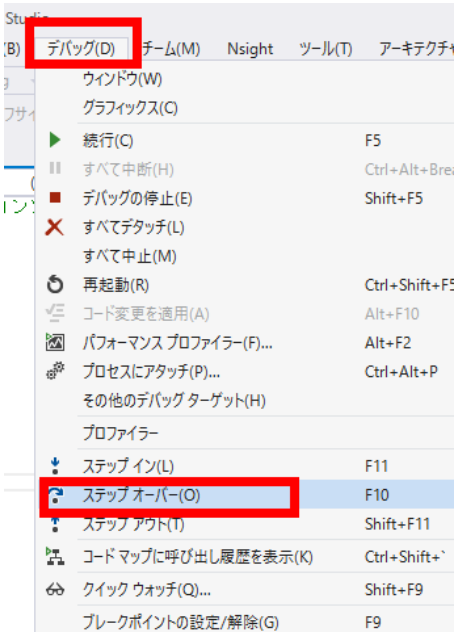


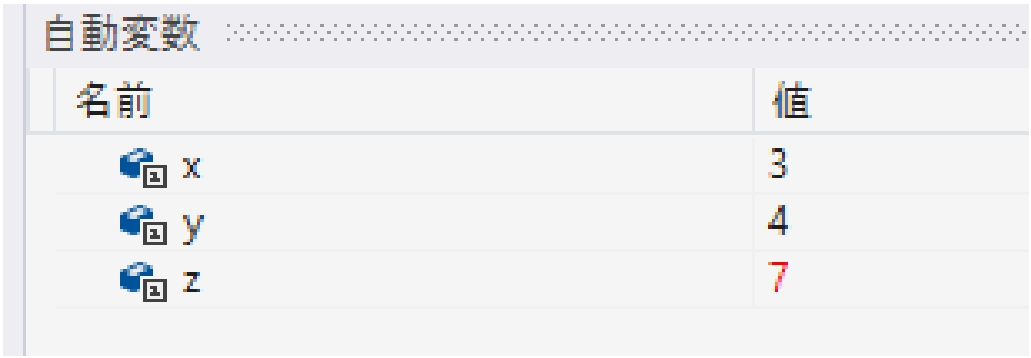
変数 y の値が 4 に変化！

「デバッグ」
→「**ステップオーバー**」
（あるいは **F10 キー**）

「**y = 4;**」が実行された

- もう１回だけステップオーバーの操作を行い、変数 z の値が 7 に変化することを確認しなさい.



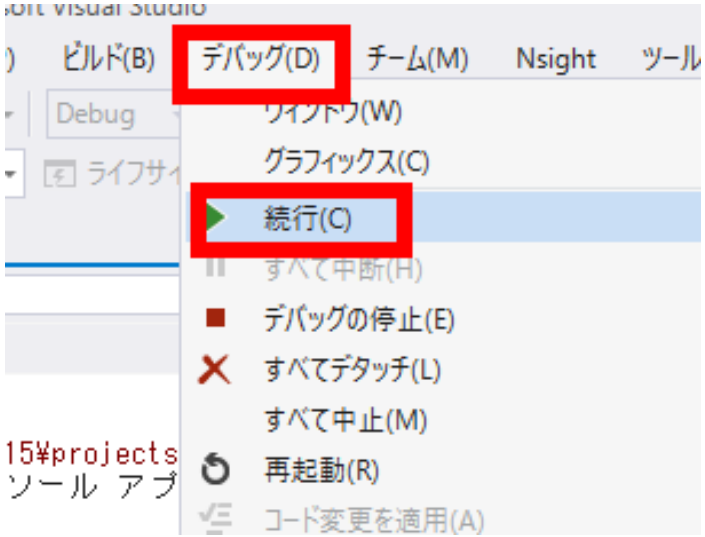


変数 z の値が 7 に変化！

「デバッグ」
→「**ステップオーバー**」
（あるいは **F10 キー**）

「**z = x + y;**」が実行された

- 最後に、プログラム実行の続行の操作を行いなさい。これで、デバッガーが終了する。



「デバッグ」
→「**続行**」

2-3 変数の変化，プログラム実行の流れ

繰り返しの例

y	x
36	3
60	5
24	2
12	1
24	2
24	2

Visual C++ のプログラム

```
static int x[6] = { 3, 5, 2, 1, 2, 2 };
static int y[6];
int i;
for (i = 0; i < 6; i++) {
    y[i] = x[i] * 12;
}
```

繰り返す処理

i の値は 0 → 1 → 2 → 3 → 4 → 5
と変化し，全部済んだら終わる

パソコン演習

- Visual Studio 2015 を起動しなさい
- Visual Studio 2015 で、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なさい

  プロジェクトの「名前」は何でもよい

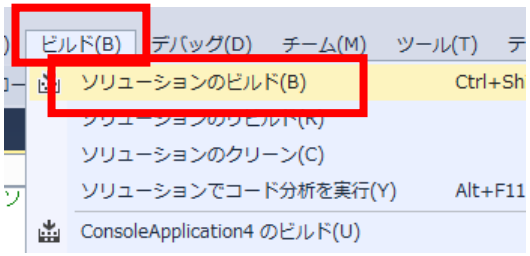
- Visual Studio 2015 のエディタを使って、ソースファイルを編集しなさい

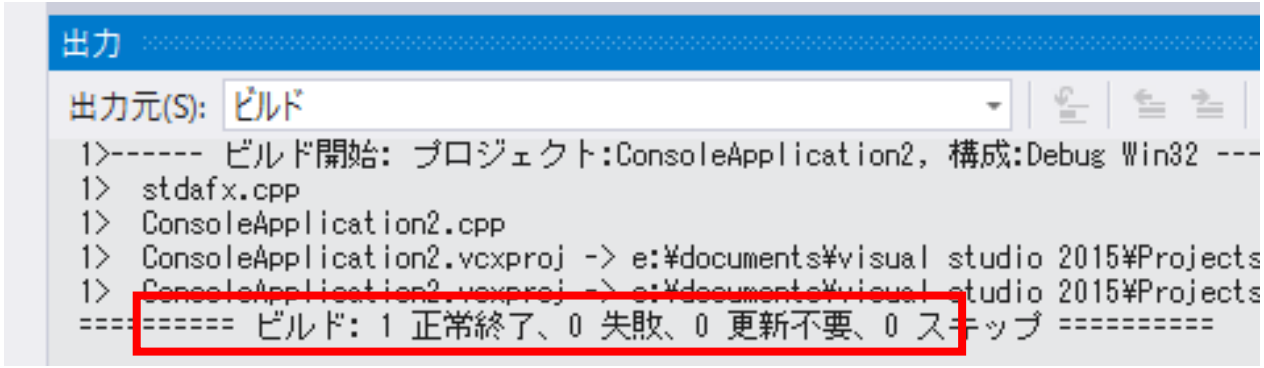
```
int main()
{
    static int x[6] = { 3, 5, 2, 1, 2, 2, };
    static int y[6];
    int i;
    for (i = 0; i < 6; i++) {
        y[i] = x[i] * 12;
    }
}
```

６行追加

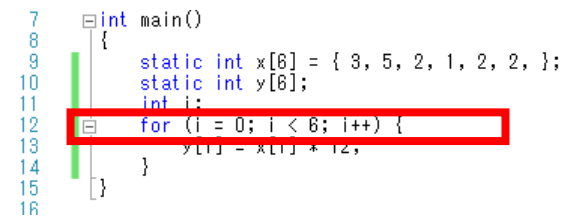
- ビルドしなさい．ビルドのあと「１　正常終了、０　失敗」の表示を確認しなさい

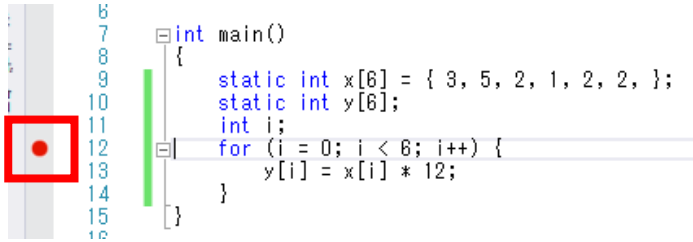
→　表示されなければ、プログラムのミスを自分で確認し、修正して、ビルドをやり直す





- Visual Studio 2015 で「for (i = 0; …」の行に、ブレークポイントを設定しなさい



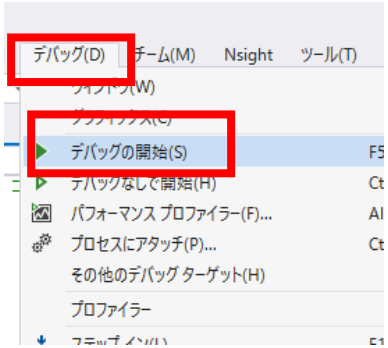


① 「**for (i = 0;**」の行をマウスでクリック

② 「デバッグ」→「**ブレークポイントの設定/解除**」

③ ブレークポイントが設定されるので確認．赤丸がブレークポイントの印

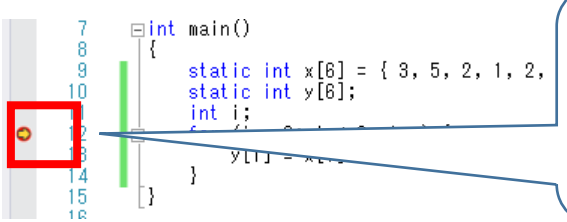
- Visual Studio 2015 で、デバッガーを起動しなさい.



「デバッグ」
→「**デバッグ開始**」

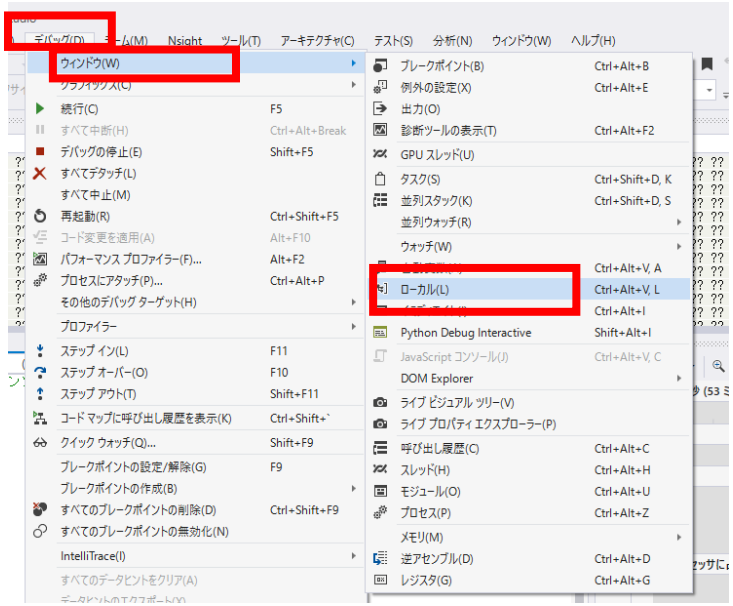
- 「for (i = 0; …」の行で、実行が中断することを確認しなさい

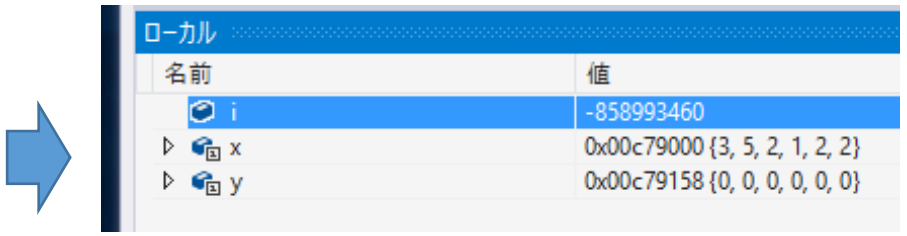
あとで使うので、中断したままにしておくこと



「for (i = 0; …」の行で実行が中断している

- 「for (i = 0; …」の行で、実行が中断した状態で、変数の値を表示させなさい．手順は次の通り．

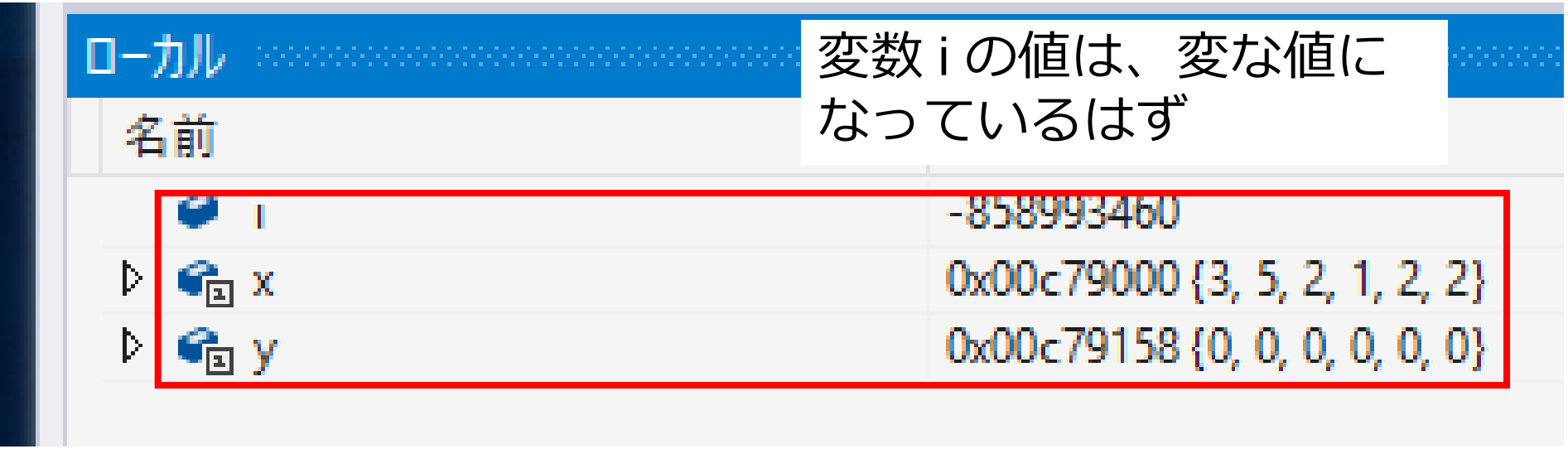




② 変数名と値の対応表が表示される

①「デバッグ」
→「ウインドウ」
→ 「**ローカル**」

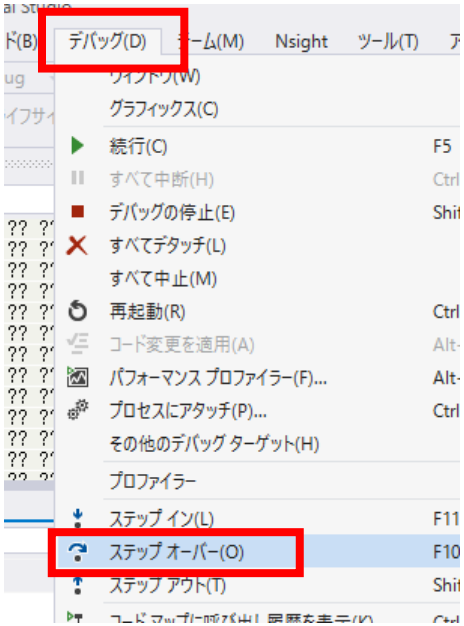
※ 次ページに拡大図



変数 i の値は、変な値になっているはず

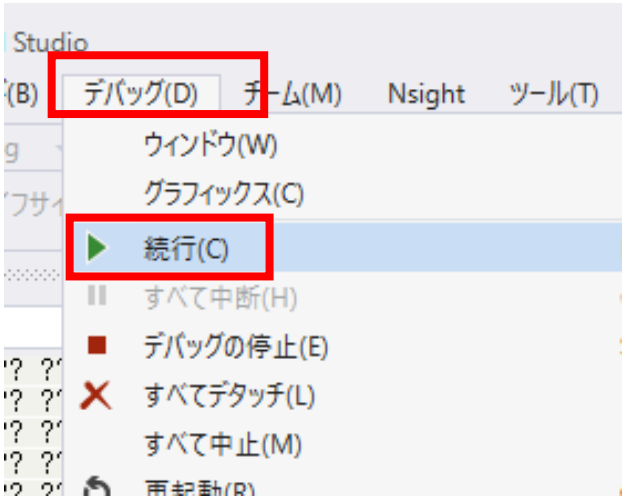
「y[i] = x[i] * 12;」は
未実行であることを確認！

- ステップオーバーの操作を１回ずつ行いながら、実行の流れを確認しなさい.



「デバッグ」
→「**ステップオーバー**」
（あるいは F10 キー）

- 最後に、プログラム実行の再開の操作を行いなさい。これで、デバッガーが終了する。



「デバッグ」
→「**続行**」

パソコン演習

- Visual Studio 2015 を起動しなさい
- Visual Studio 2015 で、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なさい

プロジェクトの「名前」は何でもよい

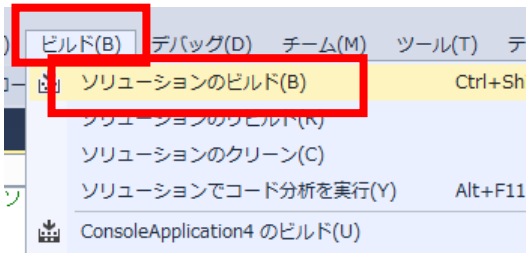
- Visual Studio 2015 のエディタを使って、ソースファイルを編集しなさい

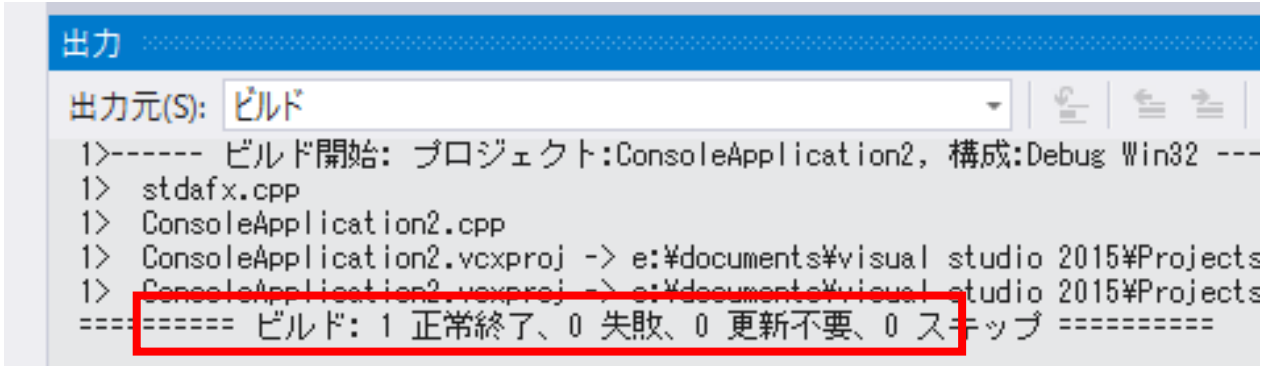
```
int main()
{
    int s, i;
    s = 0;
    for (i = 1; i <= 3; i++) {
        s = s + i;
    }
    printf("%d¥n", s);
    return 0;
}
```

← 追加

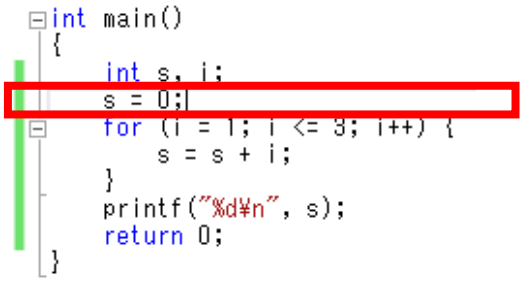
- ビルドしなさい．ビルドのあと「１　正常終了、０　失敗」の表示を確認しなさい

→　表示されなければ、プログラムのミスを自分で確認し、修正して、ビルドをやり直す

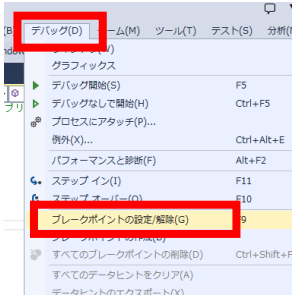




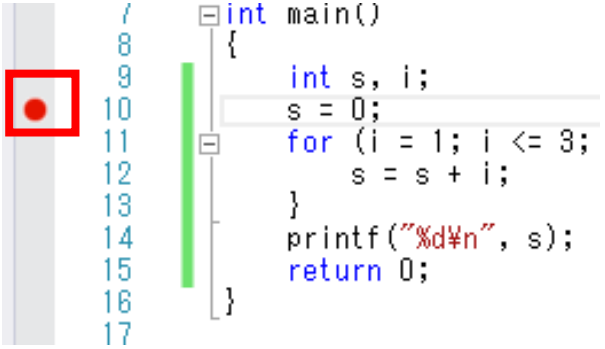
- Visual Studio 2015 で「s = 0;」の行に、ブレークポイントを設定しなさい



① 「**s = 0;**」の行をマウスでクリック

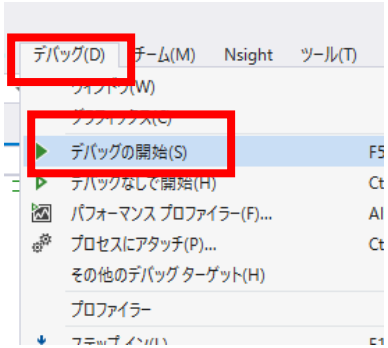


② 「デバッグ」→「**ブレークポイントの設定/解除**」



③ ブレークポイントが設定されるので確認．赤丸がブレークポイントの印

- Visual Studio 2015 で、デバッガーを起動しなさい.

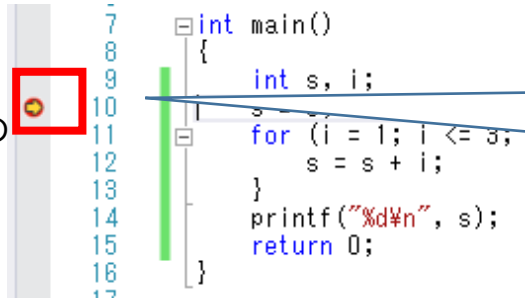


「デバッグ」
→「**デバッグの開始**」

- 「s = 0;」の行で、実行が中断することを確認しなさい

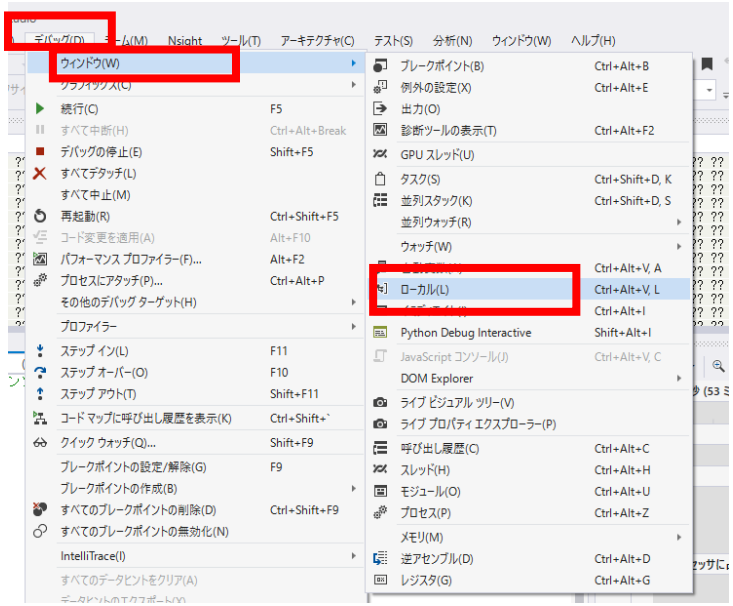
あとで使うので、中断したままにしておくこと

黄色の矢印

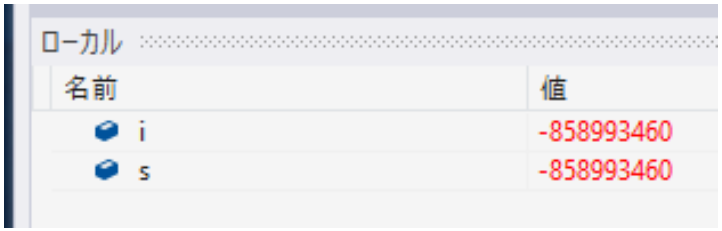


「s = 0;」の行で実行が
中断している

- 「s = 0;」の行で、実行が中断した状態で、変数の値を表示させなさい．手順は次の通り．



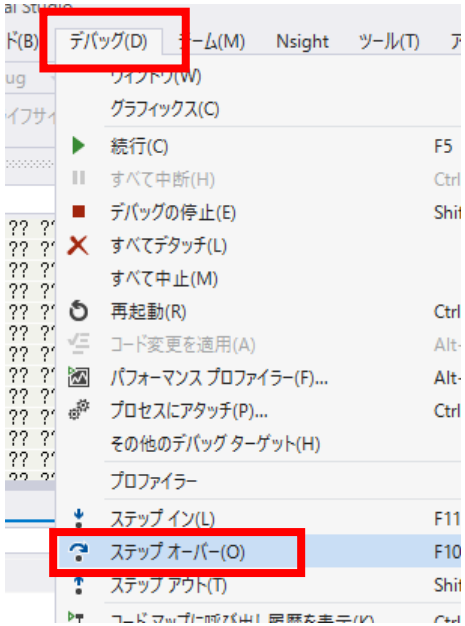
①「デバッグ」
→「ウインドウ」
→ 「**ローカル**」



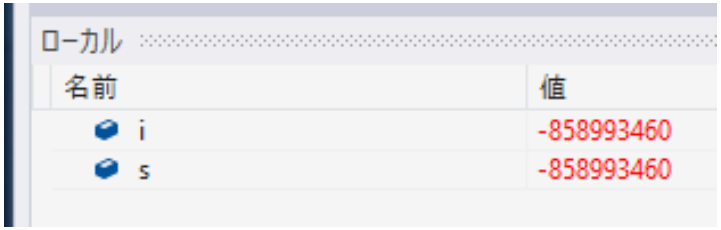
② 変数名と値の対応表が表示される

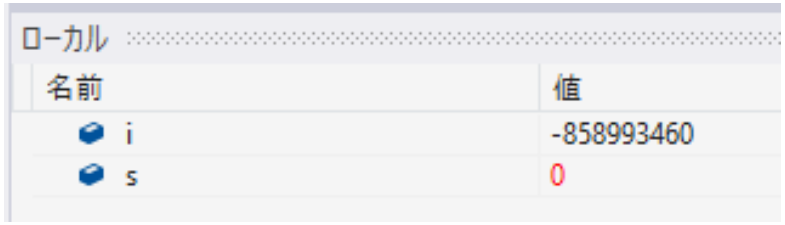
変数の値がおかしい
→ これは**正しい動作**。
今の時点では，
「s = **0;**」は未実行であることを確認！

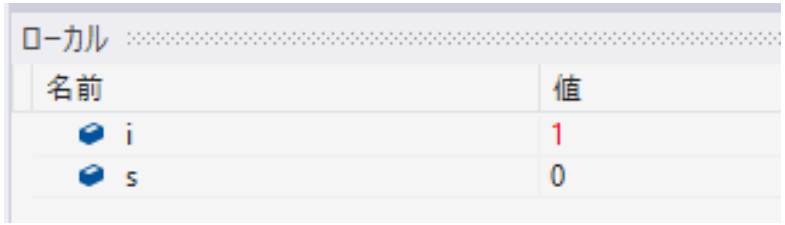
- ステップオーバーの操作を１回ずつ行いながら、実行の流れと、変数 s, i の値の変化を確認しなさい.



「デバッグ」
→「**ステップオーバー**」
（あるいは F10 キー）

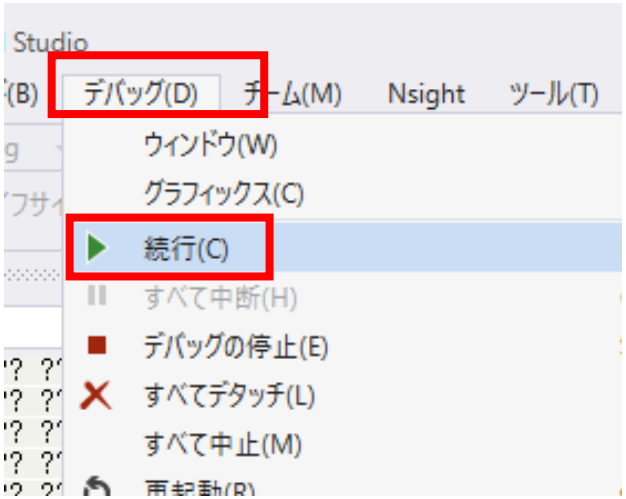

ローカル
名前
値
i
-858993460
s
-858993460


ローカル
名前
値
i
-858993460
s
0


ローカル
名前
値
i
1
s
0

さらに続ける

- 最後に、プログラム実行の再開の操作を行いなさい。これで、デバッガーが終了する。



「デバッグ」
→「**続行**」