



# vc-3. ダンプリスト, 配列

## (Visual Studio C++ の機能と操作)

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

金子邦彦



# 目次



3-1. ダンプリスト

3-2. 配列

3-3. 変数の変化, プログラム実行の流れ



# 3-1 ダンプリスト

# メモリとアドレス



- **メモリ**は**バイト**（8ビット）単位に区切られている
- 各**バイト**には0から始まる通し番号が付けられている。これが**アドレス**（番地ともいう）

メモリ内のデータは

01	00	00	00	02	00	00	00	03	00
----	----	----	----	----	----	----	----	----	----

0 1 2 3 4 5 6 7 8 9

アドレス

# ダンプリストの例



0x002F8130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0x002F813E	00	00	00	00	00	00	01	00	00	00	00	00	00	00	.....
0x002F814C	00	00	00	00	00	00	00	00	01	00	00	00	88	dd	.....h)
0x002F815A	1b	00	b0	de	1b	00	00	00	00	00	00	00	00	00	..-^.....
0x002F8168	00	00	00	00	00	00	00	00	00	00	00	00	60	91	.....
0x002F8176	28	57	00	00	00	00	00	00	00	00	00	00	00	00	(W.....
0x002F8184	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

アドレス

メモリの中身

**バイト単位**で区切られて表示.

表示は**16進数**

# ダンプリストの例



アドレスは  
0x002f8130

アドレスは  
0x002f8131

アドレスは  
0x002f8132

0x002F8130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0x002F813E	00	00	00	00	00	00	01	00	00	00	00	00	00	00	00	.....
0x002F814C	00	00	00	00	00	00	00	00	01	00	00	00	68	dd	.....h)	
0x002F815A	1b	00	b0	de	1b	00	00	00	00	00	00	00	00	00	..-^.....	
0x002F8168	00	00	00	00	00	00	00	00	00	00	00	00	60	91	.....	
0x002F8176	28	57	00	00	00	00	00	00	00	00	00	00	00	00	(W.....	
0x002F8184	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	

# クイズ



このアドレス  
は？

このアドレス  
は？

0x002F8130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0x002F813E	00	00	00	00	00	00	01	00	00	00	00	00	00	00	.....
0x002F814C	00	00	00	00	00	00	00	01	00	00	00	68	dd	.....h)	
0x002F815A	1b	00	b0	de	1b	00	00	00	00	00	00	00	00	.....)	
0x002F8168	00	00	00	00	00	00	00	00	00	00	00	60	91	.....)	
0x002F8176	28	57	00	00	00	00	00	00	00	00	00	00	00	(W.....)	
0x002F8184	00	00	00	00	00	00	00	00	00	00	00	00	00	.....)	

アドレス

メモリの中身が**バイト単位**  
で区切られて表示

メモリの中身  
を文字に置き換え  
て表示

# クイズの答え



0x002F814C

左のアドレス  
表示で分かる

0x002F8150

これは 0x002F814C に 4 足した値

0x002F8130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0x002F813E	00	00	00	00	00	00	01	00	00	00	00	00	00	00	00	.....
0x002F814C	00	00	00	00	00	00	00	01	00	00	00	68	dd	.....	h)	
0x002F8154	1b	00	00	00	00	00	00	00	00	00	00	00	00	.....	..	
0x002F8168	00	00	00	00	00	00	00	60	91	.....	.....	.....	.....	.....	.....	
0x002F8176	28	57	.....	.....	.....	.....	.....	00	00	(W	.....	.....	.....	.....	.....	
0x002F8184	00	00	.....	.....	.....	.....	.....	00	00	.....	.....	.....	.....	.....	.....	

4 つ右隣  
り

アドレス

メモリの中身が**バイト単位**  
で区切られて表示

メモリの中身  
を文字に置き換え  
て表示





## 3-2 配列



## 3-2 配列

- 配列は、同じ型の要素の並び。
- 0から始まる番号（添字）が付いている

0	1	2	3	4
8	6	4	2	3

- **コード化**されて、**メモリ**に格納されるとき、  
要素が順にメモリに格納される  
各要素のサイズは同じ

数値

(10進数)

16進数

80 →	50	00	00	00
60 →	3c	00	00	00
40 →	28	00	00	00
20 →	14	00	00	00
30 →	1e	00	00	00

4バイトの数値に**コード化**した場合

# C/C++ での配列と繰り返し



```
int main()
{
    static int x[5] = { 8, 6, 4, 2, 3 };
    static int y[5] = { 0, 0, 0, 0, 0 };
    int i;
    for (i = 0; i < 5; i++) {
        y[i] = x[i] * 10;
    }

    return 0;
}
```

繰り返す処理

i の値は 0, 1, 2, 3, 4  
と変化し, 全部済んだら終わる

- 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

```
#include "stdafx.h"
```

```
int main()  
{  
    static int x[5] = { 8, 6, 4, 2, 3 };  
    static int y[5] = { 0, 0, 0, 0, 0 };  
    int i;  
    for (i = 0; i < 5; i++) {  
        y[i] = x[i] * 10;  
    }  
    return 0;  
}
```

6行追加



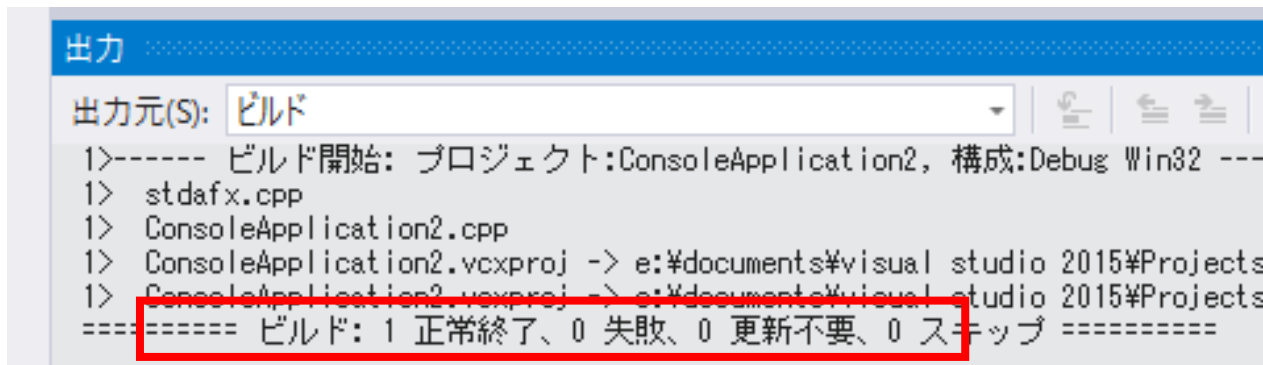
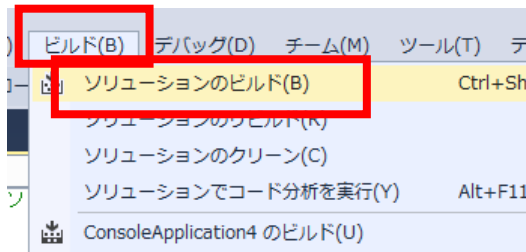
```
#include "stdafx.h"
```

```
int main()  
{  
    static int x[5] = { 8, 6, 4, 2, 3 };  
    static int y[5] = { 0, 0, 0, 0, 0 };  
    int i;  
    for (i = 0; i < 5; i++) {  
        y[i] = x[i] * 10;  
    }  
  
    return 0;  
}
```



- 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

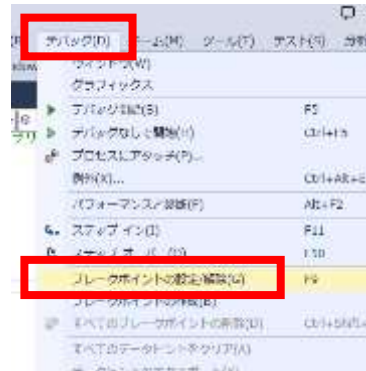
→ 表示されなければ、プログラムのミスを自分で確認し、修正して、ビルドをやり直す





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

```
int main()  
{  
    static int x[5] = { 8, 6, 4, 2, 3 };  
    static int y[5] = { 0, 0, 0, 0, 0 };  
    int i;  
    for (i = 0; i < 5; i++) {  
        y[i] = x[i] * 10;  
    }  
    return 0;  
}
```



```
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
int main()  
{  
    static int x[5] = { 8, 6, 4, 2, 3 };  
    static int y[5] = { 0, 0, 0, 0, 0 };  
    int i;  
    for (i = 0; i < 5; i++) {  
        y[i] = x[i] * 10;  
    }  
    return 0;  
}
```

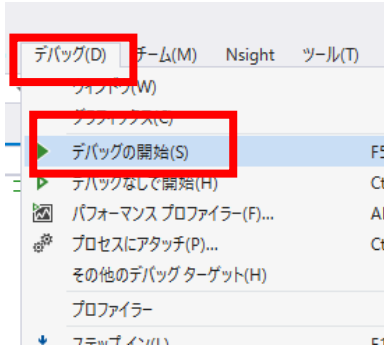
① 「int i;」の行をマウスでクリック

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」

- 「int i;」 の行で、実行が中断することを確認しなさい

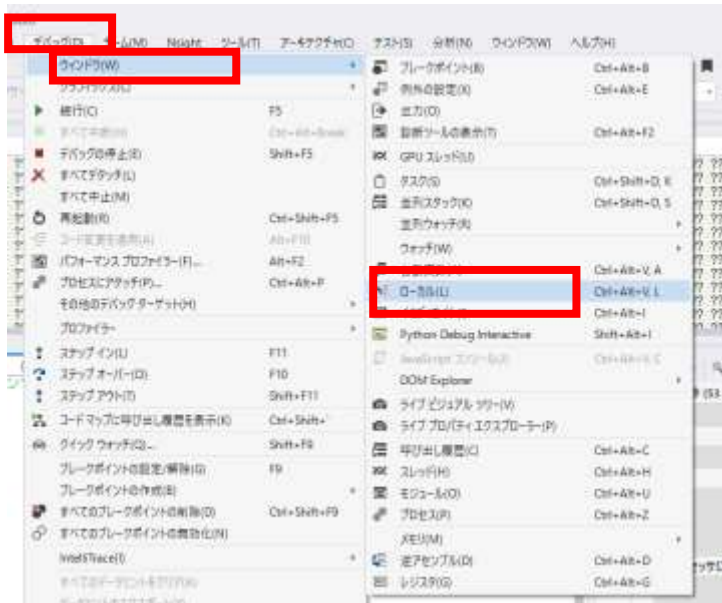
あとで使うので、中断したままにしておくこと

```
7 int main()  
8 {  
9     static int x[5] = { 8, 6, 4, 2, 0 };  
10    static int y[5] = { 0, 0, 0, 0, 0 };  
11    int i;  
12    for (i = 0; i < 5; i++) {  
13        y[i] = x[i] * 10;  
14    }  
15  
16    return 0;  
17 }  
18
```

「int i;」 の行で実行が  
中断している



- 「int i;」 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。



ローカル	
名前	値
i	-858993460
x	0x00229000 {8, 6, 4, 2, 3}
y	0x00229150 {0, 0, 0, 0, 0}

② 変数名と値の対応表が表示される




① 「デバッグ」  
→ 「ウィンドウ」  
→ 「ローカル」

※ 次ページに拡大図



## 9. ローカルウィンドウで配列 y の先頭アドレスを調べなさい

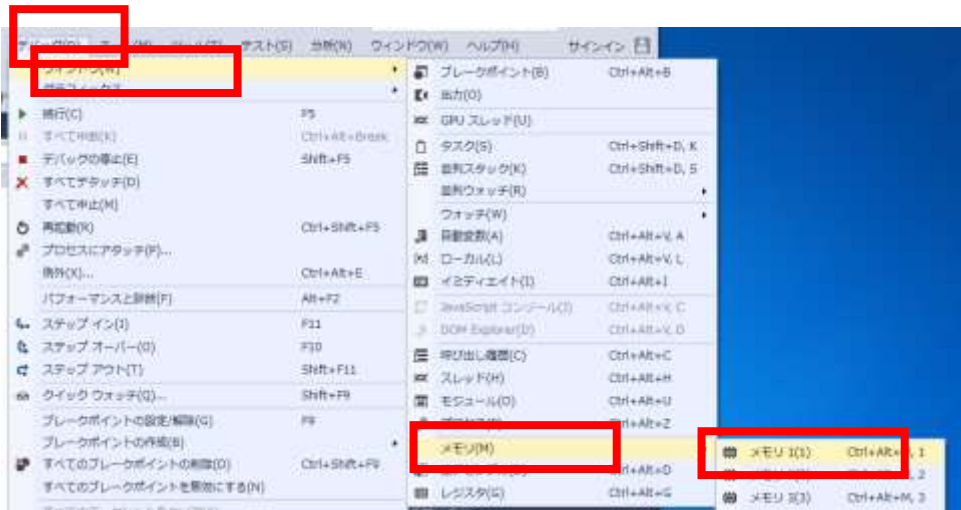
ローカル

名前	値
 i	-858993460
▷  x	0x00229000 {8, 6, 4, 2, 3}
▷  y	<b>0x00229150</b> {0, 0, 0, 0, 0}

「0x」が付いているのは16進数

y の先頭アドレスは、起動のたびに  
変化する可能性がある。

- メモリの中身を表示させなさい。手順は次の通り。



① 「デバッグ」  
→ 「ウインドウ」  
→ 「メモリ」 → 「メモリ 1」

② メモリの中身がダンプ  
リスト形式で表示される



- 配列 `y` の先頭アドレスを、  
メモリウインドウの「アドレス」のところに書き写して、  
Enter キーを押す

ローカル	名前	値
	<code>i</code>	<code>-858993460</code>
▷	<code>x</code>	<code>0x00229000 {8, 6, 4,</code>
▷	<code>y</code>	<code>0x00229150 {0, 0, 0,</code>



「`0x00229150`」のように  
頭に `0x` を付ける

メモリ 1							
アドレス:	<code>0x00229150</code>						
<code>0x00221650</code>	<code>55</code>	<code>8b</code>	<code>ec</code>	<code>81</code>	<code>ec</code>	<code>cc</code>	<code>00</code>
<code>0x0022167E</code>	<code>45</code>	<code>f8</code>	<code>83</code>	<code>7d</code>	<code>f8</code>	<code>05</code>	<code>7d</code>
<code>0x002216AC</code>	<code>cc</code>	<code>cc</code>	<code>cc</code>	<code>cc</code>	<code>cc</code>	<code>cc</code>	<code>cc</code>
<code>0x002216DA</code>	<code>cc</code>	<code>cc</code>	<code>cc</code>	<code>cc</code>	<code>cc</code>	<code>cc</code>	<code>80</code>

① 配列 `y` の先頭アドレス



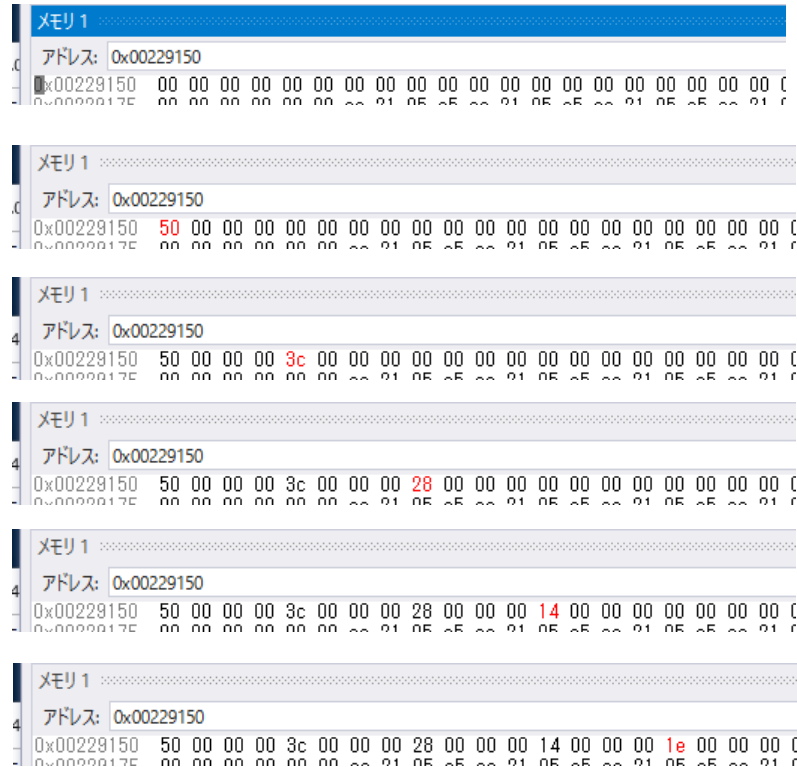
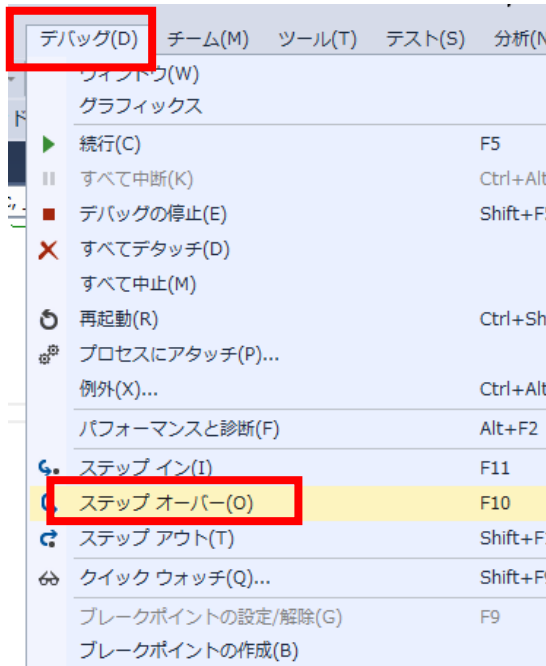
- メモリウインドウに、配列  $y$  の中身が表示されるので確認する

```
メモリ 1
アドレス: 0x00229150
0x00229150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00229175  00 00 00 00 00 00 00 00 01 05 05 00 01 05 05 00 01 05 05 00 01 05
```

00 が並んでいる



- ステップオーバーの操作を行いながら、メモリの中身の変化を確認しなさい。



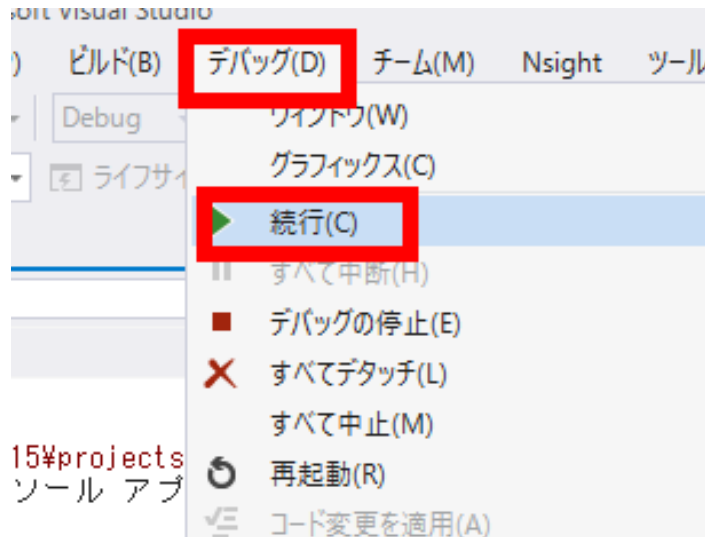
「デバッグ」  
→ 「ステップオーバー」  
(あるいは F10 キー)







- 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**」