



vc-4. 文字データと文字コード

(Visual Studio C++ の機能と操作)

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

金子邦彦



ASCII 文字コード表



コンピュータで
英数文字データを扱うときの標準

	0	1	2	3	4	5	6	7
0	NULL	DEL	SP	0	@	P		p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	(BS)	CAN	(8	H	X	h	x
9	(HT)	EM)	9	I	Y	i	y
A	(LF)	SUB	*	:	J	Z	j	z
B	(VT)	ESC	+	;	K	[k	{
C	(FF)	(FS)	,	<	L	¥	l	
D	(CR)	(GS)	-	=	M]	m	}
E	SO	(RS)	.	>	N	^	n	~
F	SI	(US)	/	?	O	_	o	DEL

太字は特別用途

- 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

```
7      int main()
8      {
9          char message[] = "hello";
10         printf("%s", message);
11         return 0;
12     }
```



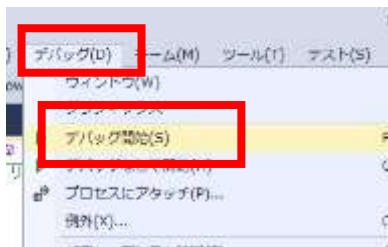
- ビルドしなさい。ビルドのあと「1 正常終了、0 失敗」の表示を確認しなさい

```
1>----- ビルド開始: プロジェクト:ConsoleApplication2, 構成:Debug Win32 -----
1> ConsoleApplication2.cpp
1> ConsoleApplication2.vcxproj -> e:%Documents%Visual Studio 2013%Projects%ConsoleApplication2%Debug%ConsoleApplication2.exe
===== ビルド: 1 正常終了、0 失敗、0 更新不要、0 スキップ =====
```

- Visual Studio 2015 で「char message = “hello”;
」の行に、ブレークポイントを設定しなさい

```
7 int main()
8 {
9 char message[] = "hello";
10 printf("%s", message);
11 return 0;
12 }
```

- 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」



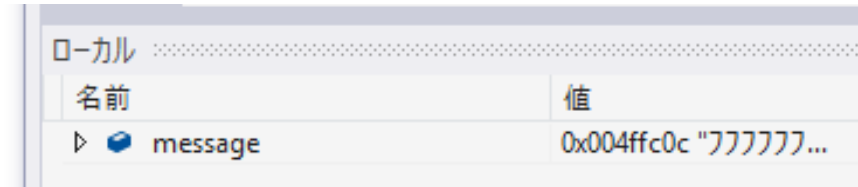
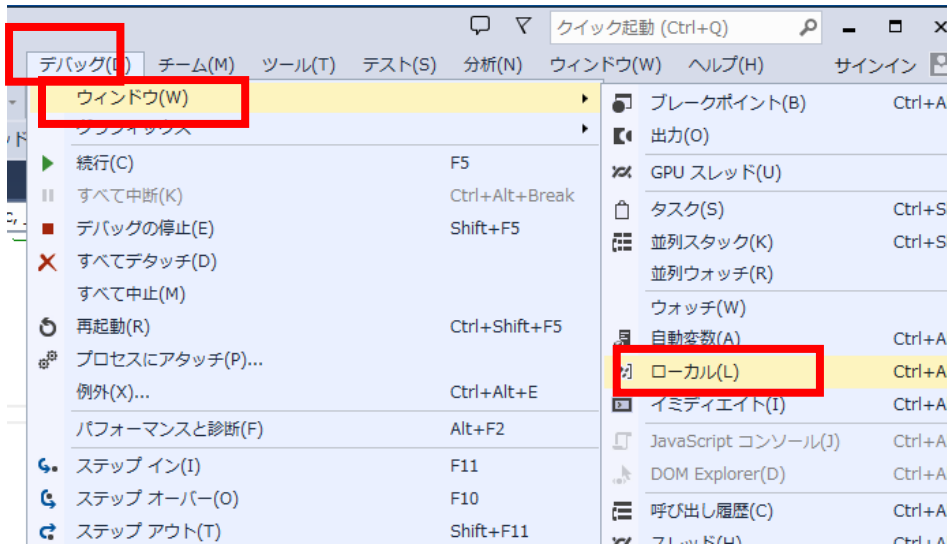
- 「char message = "hello";」 の行で、実行が中断することを確認しなさい

あとで使うので、中断したままにしておくこと

```
逆アセンブル ConsoleApplication12.cpp
ConsoleApplication12 (グローバル スコープ)
1 // ConsoleApplication12.cpp : コン
2 //
3
4 #include "stdafx.h"
5
6
7 int main()
8 {
9 char message[] = "hello";
10 printf("%s", message);
11 return 0;
12 }
```



- 「char message = "hello";」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。

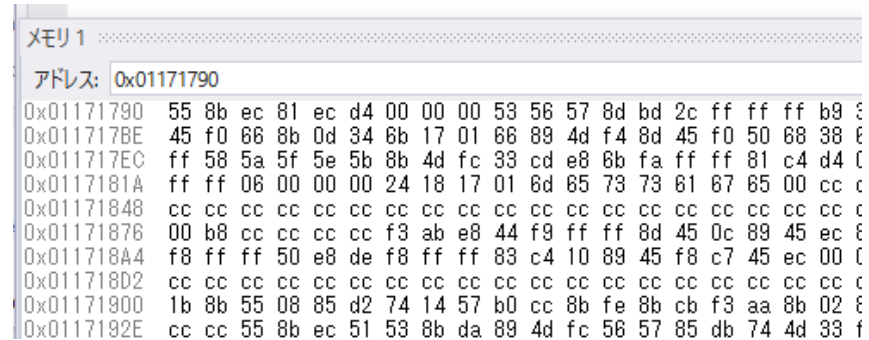
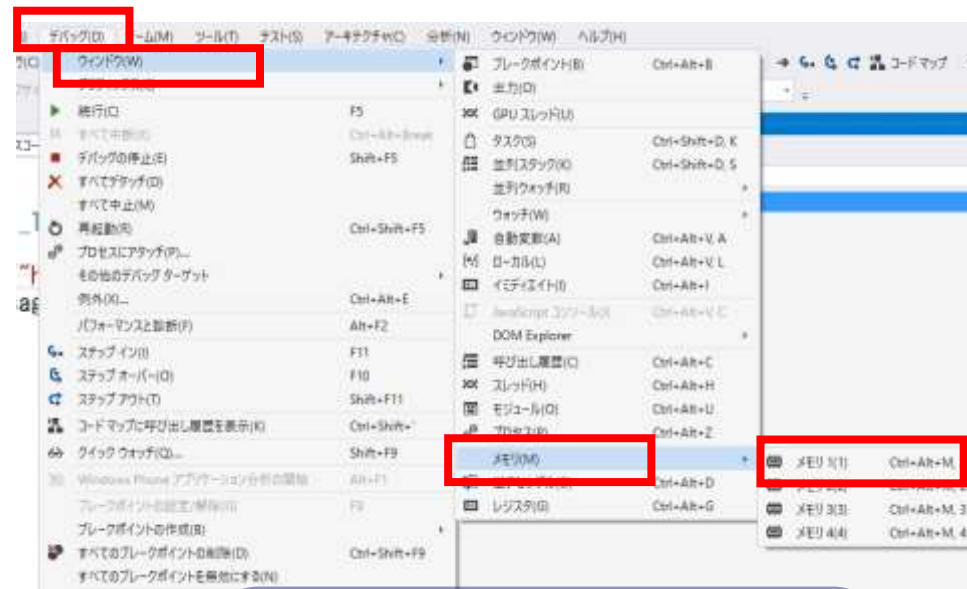


② 変数名と値の対応表が表示される

① 「デバッグ」
→ 「ウィンドウ」
→ 「ローカル」



- 「char message = "hello";」の行で、実行が中断した状態で、メモリの中身を表示させなさい。手順は次の通り。



② 「メモリ1」の画面が表示される

① 「デバッグ」
→ 「ウィンドウ」
→ 「メモリ」
→ 「メモリ1 (1)」



- 「メモリ1」の画面の「アドレス」に「&message」と入れてEnterキーを押さない

メモリ1										
アドレス: &message										
0x010FF9C8	??	??	??	??	??	??	??	??	??	??
0x010FF9F6	??	??	??	??	??	??	??	??	??	??
0x010FFA24	??	??	??	??	??	??	??	??	??	??
0x010FFA52	??	??	??	??	??	??	??	??	??	??
0x010FFA80	??	??	??	??	??	??	??	??	??	??
0x010FFAAE	??	??	??	??	??	??	??	??	??	??
0x010FFADC	??	??	??	??	??	??	??	??	??	??
0x010FFB0A	??	??	??	??	??	??	??	??	??	??
0x010FFB38	??	??	??	??	??	??	??	??	??	??

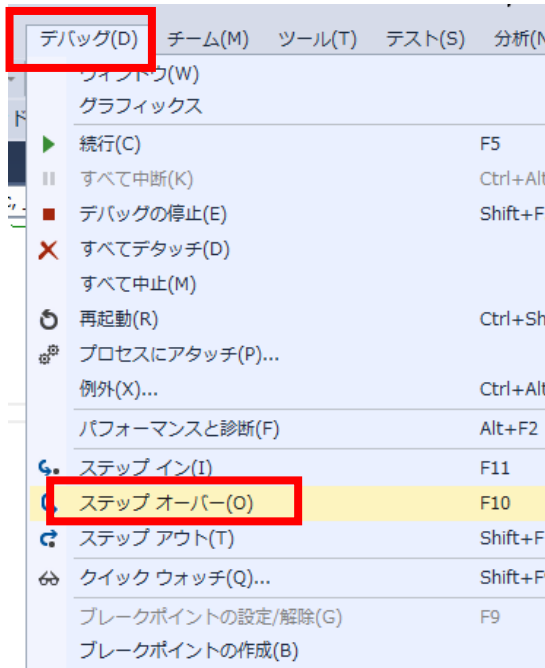


メモリ1										
アドレス: 0x004FFD84										
0x004FFD84	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x004FFDB2	45	b1	46	10	0b	01	46	10	0b	01
0x004FFDE0	80	95	0b	01	00	00	00	00	b0	fd
0x004FFDEE	0b	01	24	fe	4f	00	84	84	80	74
0x004FFE3C	00	60	20	00	00	00	00	00	00	00
0x004FFE6A	00	00	7c	fe	4f	00	ba	2f	b7	77
0x004FFE98	00	00	00	00	00	00	00	00	00	00
0x004FFEC6	00	00	00	00	00	00	00	00	00	00
0x004FFFF4	nn	nn	nn	nn	nn	nn	nn	nn	nn	nn

画面が変化するので確認する



- ステップオーバーの操作を1回行いなさい



メモリ 1

アドレス: 0x004FFD84

0x004FFD84	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	d7	ea
0x004FFDB2	45	b1	46	10	0b	01	46	10	0b	01	00	60	20	00
0x004FFDE0	80	95	0b	01	00	00	00	00	b0	fd	4f	00	00	00



char message[] = "hello";

メモリ 1

アドレス: 0x004FFD84

0x004FFD84	68	65	6c	6c	6f	00	cc	cc	cc	cc	cc	cc	d7	ea
0x004FFDB2	45	b1	46	10	0b	01	46	10	0b	01	00	60	20	00
0x004FFDE0	80	95	0b	01	00	00	00	00	b0	fd	4f	00	00	00

「デバッグ」
→ 「ステップオーバー」
(あるいは F10 キー)



メモリ

アドレス: 0x004FFD84

0x004FFD84	68	65	6c	6c	6f	00	cc	cc	cc	cc	cc	cc	d7	ea
0x004FFDB2	45	b1	46	10	0b	01	46	10	0b	01	00	60	20	00
0x004FFDE0	80	95	0b	01	00	00	00	00	b0	fd	4f	00	00	00

メモリの中身は **68 65 6c 6c 6f 00** に変化

68→h

65→e

6c→l

6c→l

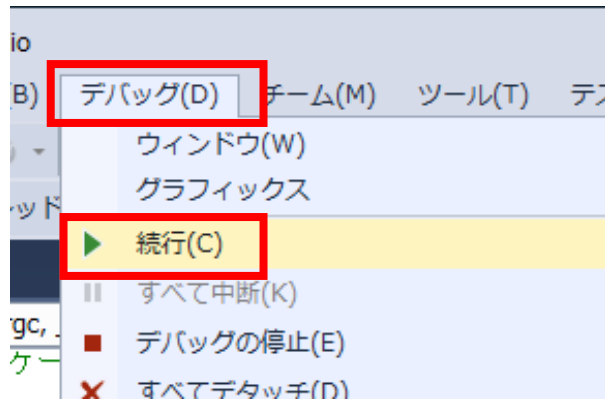
6f→o

ASCIIコード

00→文字列の終わり



- 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」



- プログラムを次のように書き換えて、同じことをもう1度行いなさい

```
7  
8  
9  
10  
11  
12  
13 ^
```

```
int main()  
{  
    char message[] = "aaa,bb cc";  
    printf("%s", message);  
    return 0;  
}
```

メモリ

アドレス: 0x00F5FA28

0x00F5FA28	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x00F5FA58	a0	00	0a	35	7c	e9	46	10	a0	00	46	



メモリ

アドレス: 0x00F5FA28

0x00F5FA28	61	61	61	2c	62	62	20	63	63	00	cc	
0x00F5FA58	a0	00	0a	35	7c	e9	46	10	a0	00	46	