Program Patterns: Comments & Debugging

YoungWoon Cha

Fall 2022

Roadmap

- Commenting
- Debugging

Commenting

Source: https://www.cs.utah.edu/~germain/PPS/Topics/commenting.html

Commenting

- **Human Readable Descriptions**
  - Proper use of commenting can make code maintenance much easier, as well as helping make finding bugs faster.
  - Further, commenting is very important when writing functions that other people will use.
  - Remember, well documented code is as important as correctly working code.

Summary of comment types

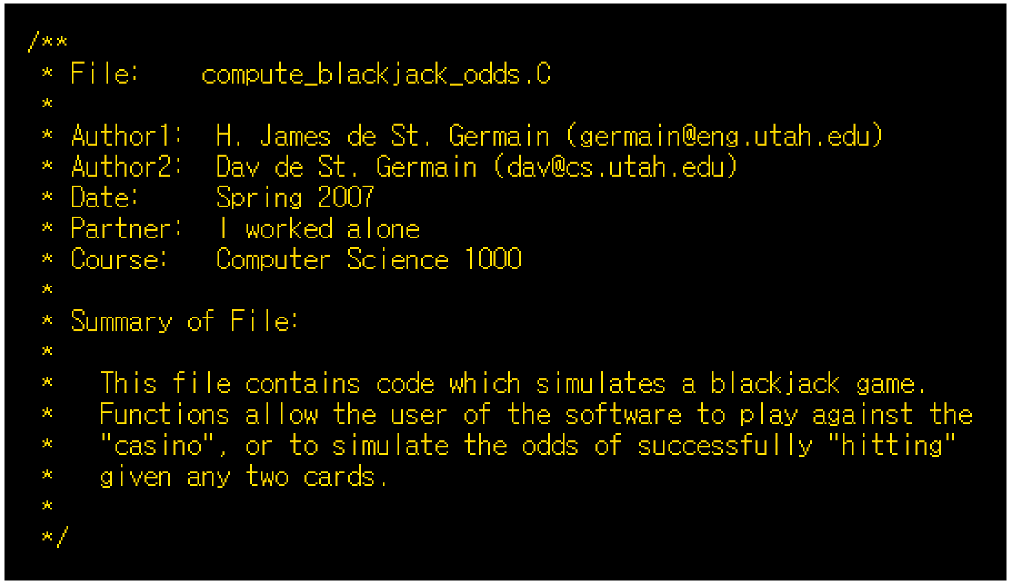
- Single line comment
  - // for a single line.
- Block comment
  - /*   for multiple lines   */

Where to Comment

- The top of any program file.
  - **"Header Comment"**
- Above every function.
  - function header
  - "sub-component"
- In line
  - Any "tricky" code

The top of any program file.

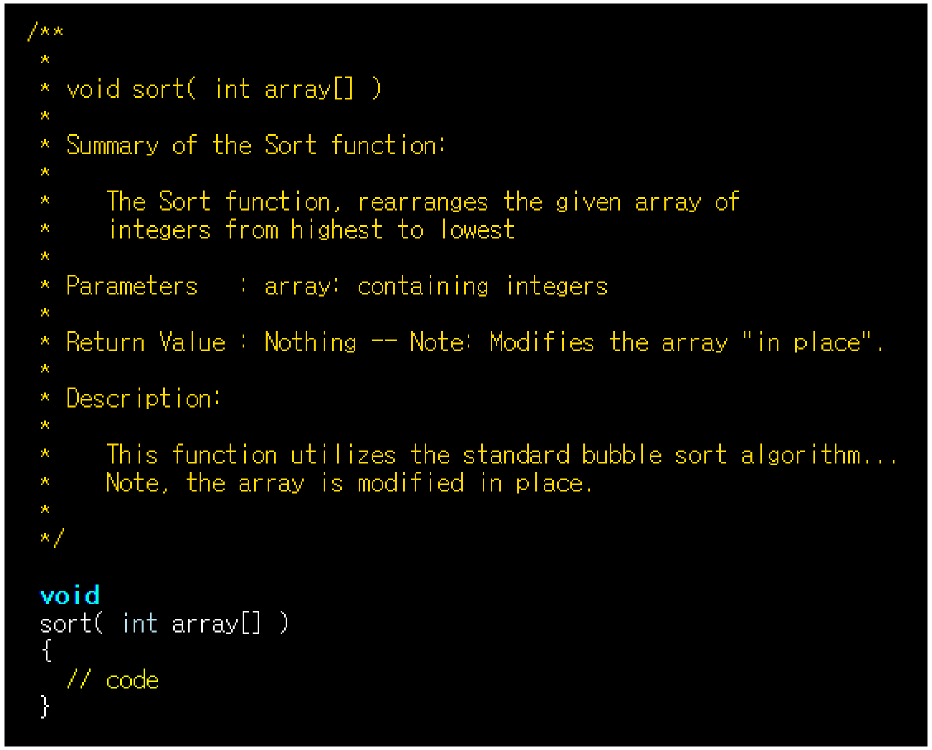
- This is called the **"Header Comment"**.
- It should include all the defining information about
  - The author, date, and course number.
  - A description of what the code in the file accomplishes
  - A list of any modifications (bug fixes) to the file. Note this is not as important for programs written in class, but important in the real world.



```
/**
 * File:    compute_blackjack_odds.C
 *
 * Author1:  H. James de St. Germain (germain@eng.utah.edu)
 * Author2:  Dav de St. Germain (dav@cs.utah.edu)
 * Date:     Spring 2007
 * Partner:  I worked alone
 * Course:   Computer Science 1000
 *
 * Summary of File:
 *
 *   This file contains code which simulates a blackjack game.
 *   Functions allow the user of the software to play against the
 *   "casino", or to simulate the odds of successfully "hitting"
 *   given any two cards.
 *
 */
```

Above every function.

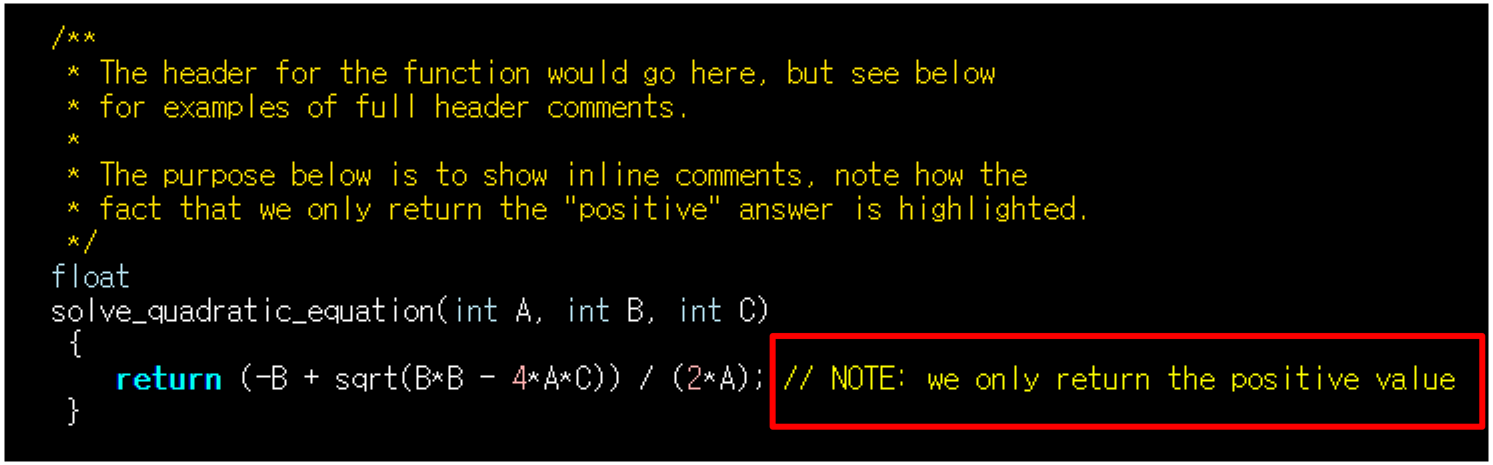
- This is called the function header and provides information about the purpose of this "sub-component" of the program.
- Every function must have a separate header comment.
- When and if there is only one function in a file, the function header and file header comments should be merged into a single comment.



```
/**
 *
 * void sort( int array[] )
 *
 * Summary of the Sort function:
 *
 *    The Sort function, rearranges the given array of
 *    integers from highest to lowest
 *
 * Parameters   : array: containing integers
 *
 * Return Value : Nothing -- Note: Modifies the array "in place".
 *
 * Description:
 *
 *    This function utilizes the standard bubble sort algorithm...
 *    Note, the array is modified in place.
 *
 */

 void
 sort( int array[] )
 {
   // code
 }
```

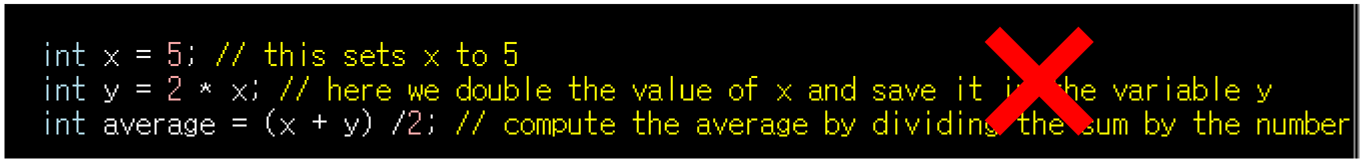
In line

- Any "tricky" code where it is not immediately obvious what you are trying to accomplish, should have comments right above it or on the same line with it.



```
/**
 * The header for the function would go here, but see below
 * for examples of full header comments.
 *
 * The purpose below is to show inline comments, note how the
 * fact that we only return the "positive" answer is highlighted.
 */
float
solve_quadratic_equation(int A, int B, int C)
 {
    return (-B + sqrt(B*B - 4*A*C)) / (2*A); // NOTE: we only return the positive value
 }
```

- They should not restate something that is "obvious".



```
int x = 5; // this sets x to 5
int y = 2 * x; // here we double the value of x and save it in the variable y
int average = (x + y) /2; // compute the average by dividing the sum by the number
```

Self Documenting Code

- Self documenting code uses well chosen variable names (and function names) to make the code read as close to English as possible.
- This should be your goal.

- Which one is a better description?
  - float g; vs. float gravity;
  - float a = (x+y)/2; vs. float average = (x+y)/2;

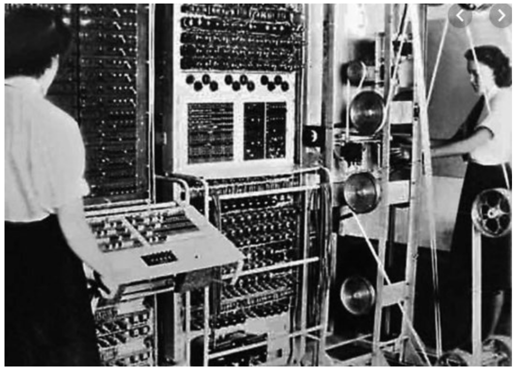
Software Programming Pattern

Debugger (based on Visual Studio)

Debugging

- Debugging is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system.

The terms "bug" and "debugging" are popularly attributed to Admiral Grace Hopper in the 1940s.[1] While she was working on a Mark II computer at Harvard University, her associates discovered a moth stuck in a relay and thereby impeding operation, whereupon she remarked that they were "debugging" the system.


Mark II PC

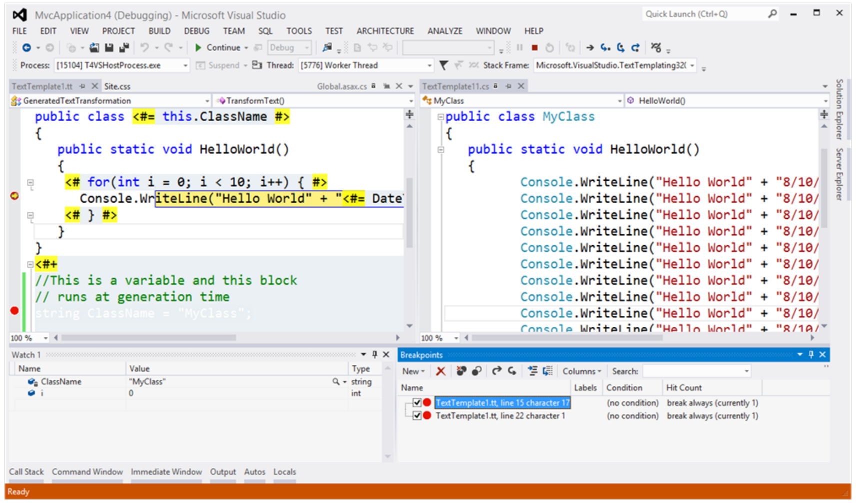
Types of program error

- Compile errors (syntax errors)
  - Typo
  - Violation of language rules
- Runtime errors
  - Memory allocation
  - Stack overflow
- Logic errors
  - Wrong answer

Types of debugger

- GUI type
  - Provide a graphical debugging environment
  - Usually, built in IDE (Visual studio, Eclipse, PyCharm, XCode)
- Text type
  - Based on text input/output (command line method)
  - Run independently (But, Some GUI-wrapped applications.)

Type of debugger


MvcApplication4 (Debugging) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Continue
Debug
Process: [15104] T4VSHostProcess.exe
Suspend
Thread: [5776] Worker Thread
Stack Frame: Microsoft.VisualStudio.TextTemplating32(
TextTemplate1.tt
Site.css
Global.asax.cs
TextTemplate11.cs
GeneratedTextTransformation
TransformText()
MyClass
HelloWorld()
Solution Explorer
Server Explorer
100 %
Call Stack
Command Window
Immediate Window
Output
Autos
Locals
Ready

```
public class <#= this.ClassName #>
{
    public static void HelloWorld()
    {
     <# for(int i = 0; i < 10; i++) { #>
        Console.WriteLine("Hello World" + "<#= Date
     <# } #>
    }
}
<#+
//This is a variable and this block
// runs at generation time
string ClassName = "MyClass";
```

```
public class MyClass
{
    public static void HelloWorld()
    {
            Console.WriteLine("Hello World" + "8/10/
            Console.WriteLine("Hello World" + "8/10/
            Console.WriteLine("Hello World" + "8/10/
            Console.WriteLine("Hello World" + "8/10/
            Console.WriteLine("Hello World" + "8/10/
            Console.WriteLine("Hello World" + "8/10/
            Console.WriteLine("Hello World" + "8/10/
            Console.WriteLine("Hello World" + "8/10/
            Console.WriteLine("Hello World" + "8/10/
            Console.WriteLine("Hello World" + "8/10/
```

Watch 1

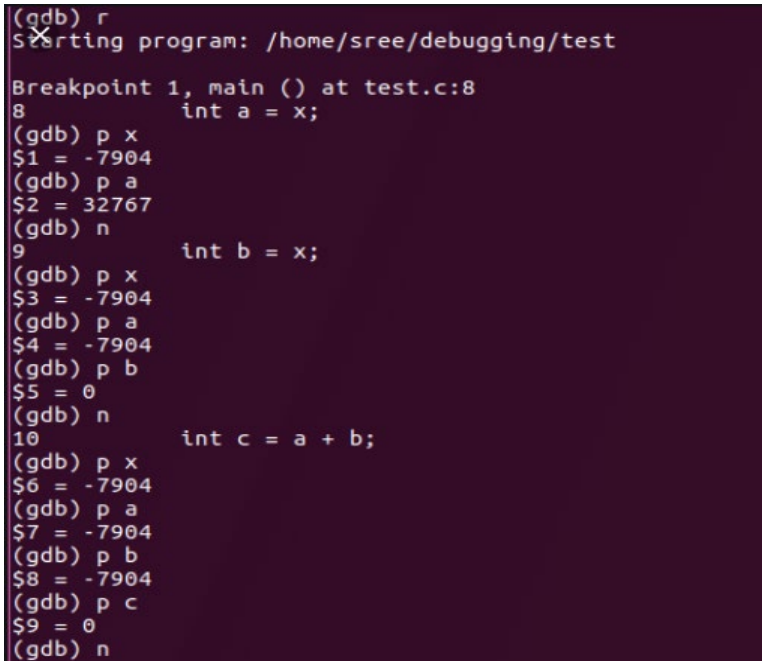
Name	Value	Type
ClassName	"MyClass"	string
i	0	int

Breakpoints

New ▾ Columns ▾ Search:

Name	Labels	Condition	Hit Count
☑ ● TextTemplate1.tt, line 15 character 17		(no condition)	break always (currently 1)
☑ ● TextTemplate1.tt, line 22 character 1		(no condition)	break always (currently 1)

Visual Studio
Debugger



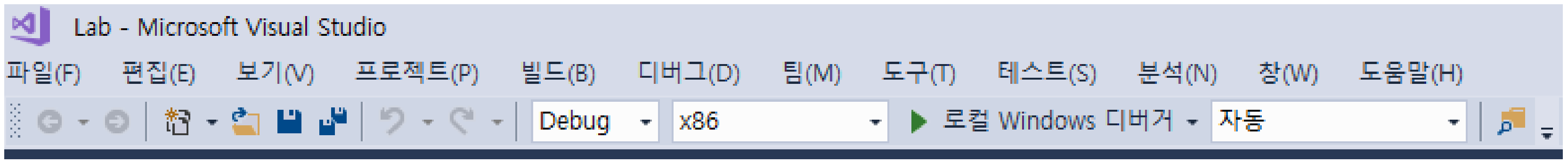
```
(gdb) r
Starting program: /home/sree/debugging/test

Breakpoint 1, main () at test.c:8
8           int a = x;
(gdb) p x
$1 = -7904
(gdb) p a
$2 = 32767
(gdb) n
9           int b = x;
(gdb) p x
$3 = -7904
(gdb) p a
$4 = -7904
(gdb) p b
$5 = 0
(gdb) n
10          int c = a + b;
(gdb) p x
$6 = -7904
(gdb) p a
$7 = -7904
(gdb) p b
$8 = -7904
(gdb) p c
$9 = 0
(gdb) n
```

GDB

Compile mode (Visual C++)

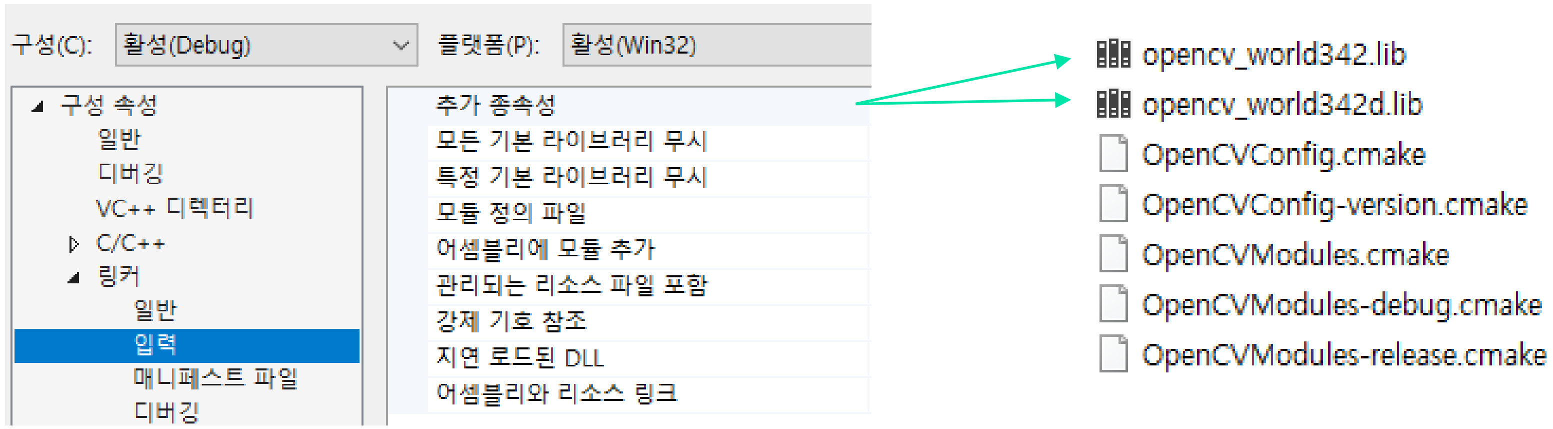
- Debug produces
  - Binary code
  - Additional Info
  - Poor optimization performance

- Release
  - Binary Code only


Lab - Microsoft Visual Studio

Compile mode (Visual C++)

- Caution
  - When you change compile mode, you should modify external libraries.



구성(C): 활성(Debug) 플랫폼(P): 활성(Win32)

- 구성 속성
  - 일반
  - 디버깅
  - VC++ 디렉터리
  - C/C++
  - 링커
    - 일반
    - 입력
    - 매니페스트 파일
    - 디버깅

추가 종속성
모든 기본 라이브러리 무시
특정 기본 라이브러리 무시
모듈 정의 파일
어셈블리에 모듈 추가
관리되는 리소스 파일 포함
강제 기호 참조
지연 로드된 DLL
어셈블리와 리소스 링크

opencv_world342.lib
opencv_world342d.lib
OpenCVConfig.cmake
OpenCVConfig-version.cmake
OpenCVModules.cmake
OpenCVModules-debug.cmake
OpenCVModules-release.cmake

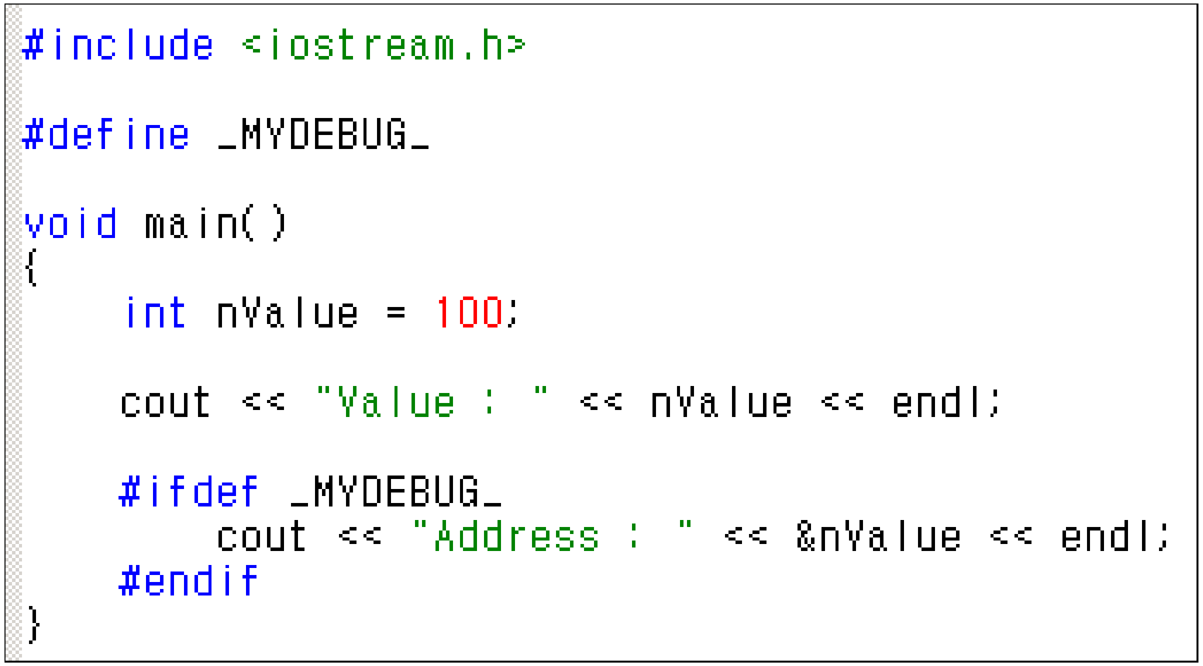
Debugging in Code

- Using preprocessor
  - #define, #undef
  - #if, #elif, #else, #endif
  - #ifdef, #ifndef

- Step 1: Set a flag for debugging
- Step 2: Locate debugging code into a block, wrapping by preprocessor
- Step 3: After debugging, remove a flag or set it False

Debugging in Code

- Example

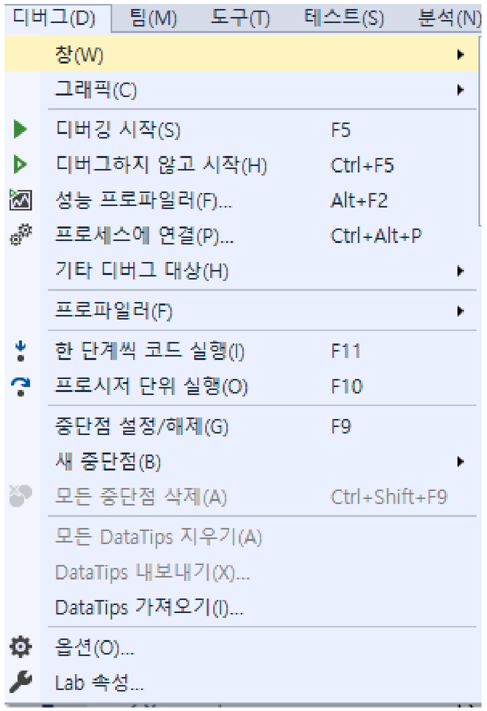


```
#include <iostream.h>

#define _MYDEBUG_

void main( )
{
	int nValue = 100;

	cout << "Value :  " << nValue << endl;

	#ifdef _MYDEBUG_
		cout << "Address :  " << &nValue << endl;
	#endif
}
```

Debugging using Tools

- How to use a debugging tool?
  - There is no special way, just run in debugging mode.
  - F5 key (Visual Studio 201X)
  - Maybe some of you already run program as debug mode.

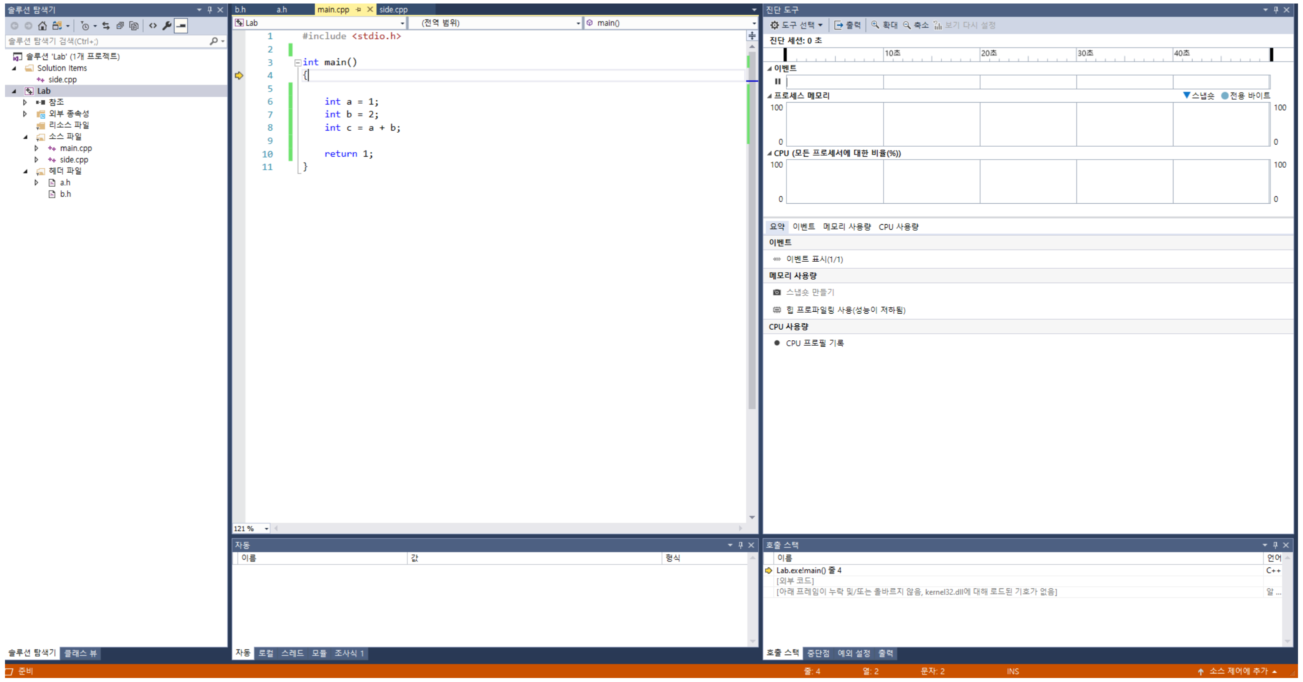
  - If the debugging is paused, the configuration changes to the debugging environment.



디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N)
창(W)
그래픽(C)
디버깅 시작(S) F5
디버그하지 않고 시작(H) Ctrl+F5
성능 프로파일러(F)... Alt+F2
프로세스에 연결(P)... Ctrl+Alt+P
기타 디버그 대상(H)
프로파일러(F)
한 단계씩 코드 실행(I) F11
프로시저 단위 실행(O) F10
중단점 설정/해제(G) F9
새 중단점(B)
모든 중단점 삭제(A) Ctrl+Shift+F9
모든 DataTips 지우기(A)
DataTips 내보내기(X)...
DataTips 가져오기(I)...
옵션(O)...
Lab 속성...

Debugging using Tools

- Debugging mode



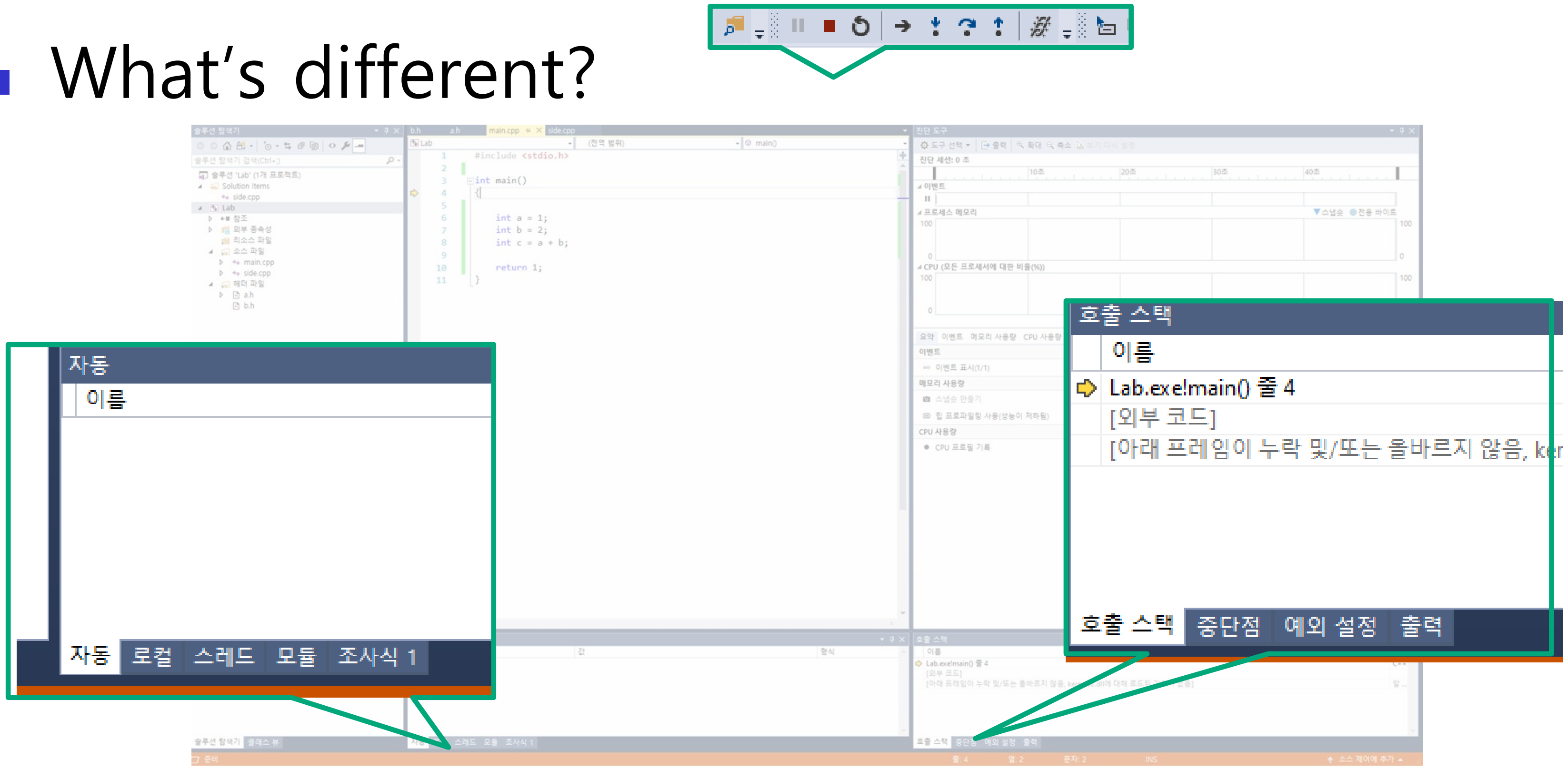
```
#include <stdio.h>

int main()
{


    int a = 1;
    int b = 2;
    int c = a + b;

    return 1;
}
```

이름	값	형식

이름	언어
Lab.exe!main() 줄 4	C++
[외부 코드]	
[아래 프레임이 누락 및/또는 올바르지 않음, kernel32.dll에 대해 로드된 기호가 없음]	알 ...

Debugging using Tools

- What's different?



```
#include <stdio.h>

int main()
{

    int a = 1;
    int b = 2;
    int c = a + b;

    return 1;
}
```

호출 스택

이름
Lab.exe!main() 줄 4
[외부 코드]
[아래 프레임이 누락 및/또는 올바르지 않음, ker

Debugging start

- Several options
  - F11: Run code in debugging mode at the entrance point.
  - F10: Run code in debugging mode at the first procedure
  - F5: Run code in debugging mode until it meets a breaking point

Exercise

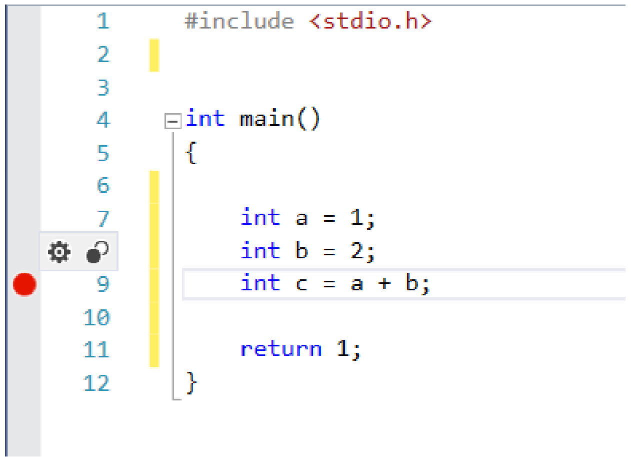
1. Run by F11
2. Run by F10
3. Run by F5

```
#include <stdio.h>

int main()
{
    int a = 1;
    int b = 2;
    int c = a + b;

return 1;
}
```

Debugging skill

- Break Point
  - Several break points can be set.
  - When running in debugging mode, execution pauses when it encounters a breaking point.
  - BP can be set in lines.

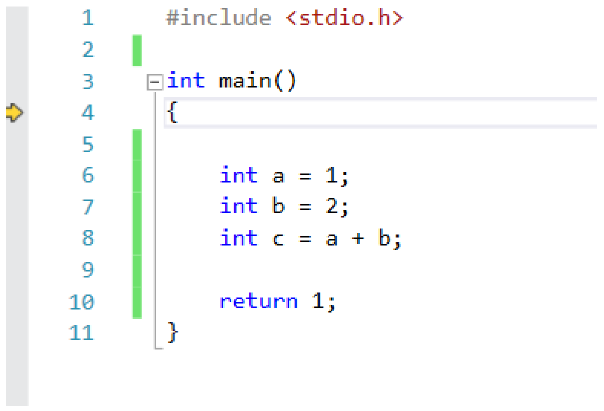
  - Method: F9 key or Mouse L-button click on left size of line numbers.



```
#include <stdio.h>


int main()
{

    int a = 1;
    int b = 2;
    int c = a + b;

    return 1;
}
```

Debugging start

- Debugging point
  - Represented as a yellow arrow
  - It means running point stops at there.



```
#include <stdio.h>

int main()
{

    int a = 1;
    int b = 2;
    int c = a + b;

    return 1;
}
```

Example

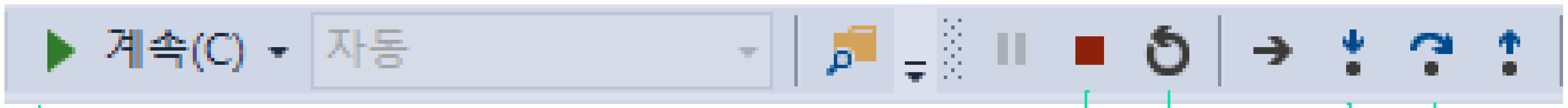
```
#include <stdio.h>

int sum( int a, int b)
{
    int c = 0;
    c = a + b;
    return c;
}

int main()
{
    int a = 10;
    int b = 20;
●   int s = sum(a, b);
    printf("sum
    = %d₩n", s);

    return 1;
}
```

- Set a break point at the line of “int s = sum(a, b);
- Run as debug mode

Example

- The debugging process pauses at the first break point
- What can we do next??
  - Stop debugging (Shift + F5)
  - Restart running (Ctrl + Shift + F5)
  - Just Go
  - Step into procedure (F11)
  - Step over (F10)
  - Step out from the procedure (Shift + F11)
- You can consider a “procedure” as a function or just a command

Example



- Stop debugging
- Restart running
- Just Go
- Step into procedure
- Step over
- Step out from the procedure

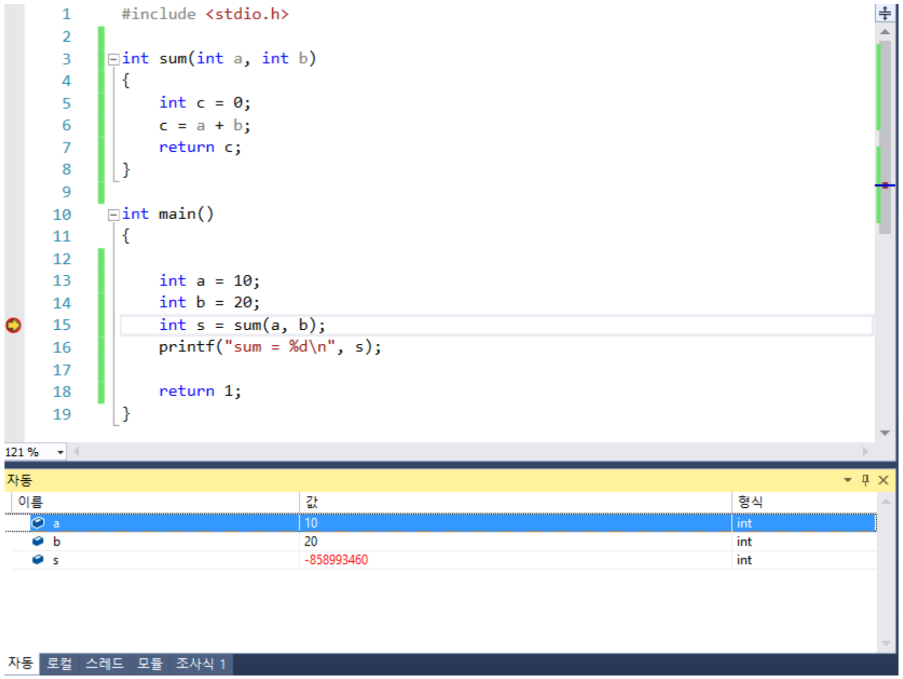
Exercise

```
#include <stdio.h>

int sum( int a, int b)
{
    int c = 0;
    c = a + b;
    return c;
}

int main()
{
    int a = 10;
    int b = 20;
●   int s = sum(a, b);
    printf("sum
    = %d₩n", s);

    return 1;
}
```

- Exercise
- Select six different ways at the first break point
  - Stop debugging
  - Restart running
  - Just Go
  - Step into procedure
  - Step over
  - Step out from the procedure

Parameters Monitoring

- You can see current values of variables in debugging mode



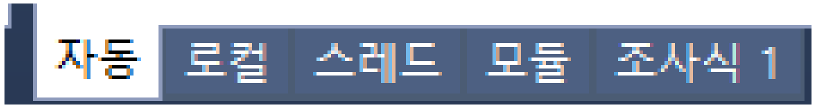
```c
#include <stdio.h>

int sum(int a, int b)
{
    int c = 0;
    c = a + b;
    return c;
}

int main()
{

    int a = 10;
    int b = 20;
    int s = sum(a, b);
    printf("sum = %d\n", s);

    return 1;
}
```

121 %

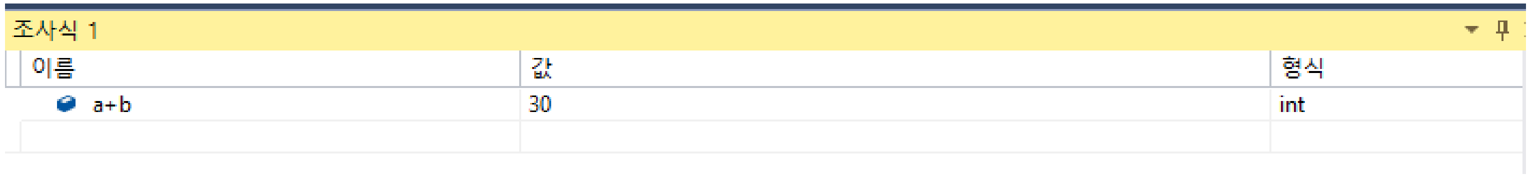
자동

이름	값	형식
a	10	int
b	20	int
s	-858993460	int

Parameters Monitoring



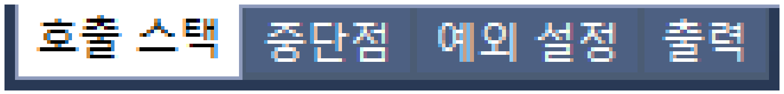
- **자동: show variables related with current position**
- **로컬: show all local variables in the function**
- 스레드: thread information
- 모듈: information about binary code and runtime library(dll)
- **조사식: any equation you made**



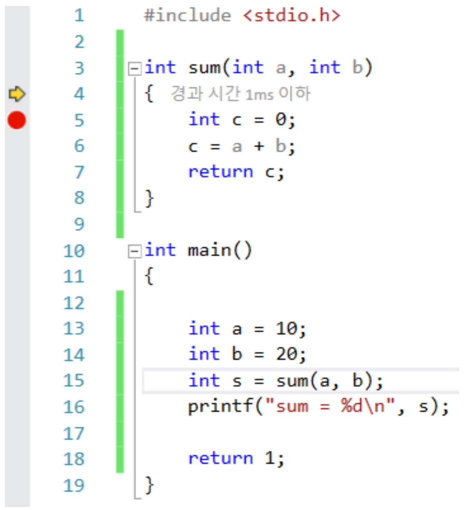
조사식 1

이름	값	형식
a+b	30	int

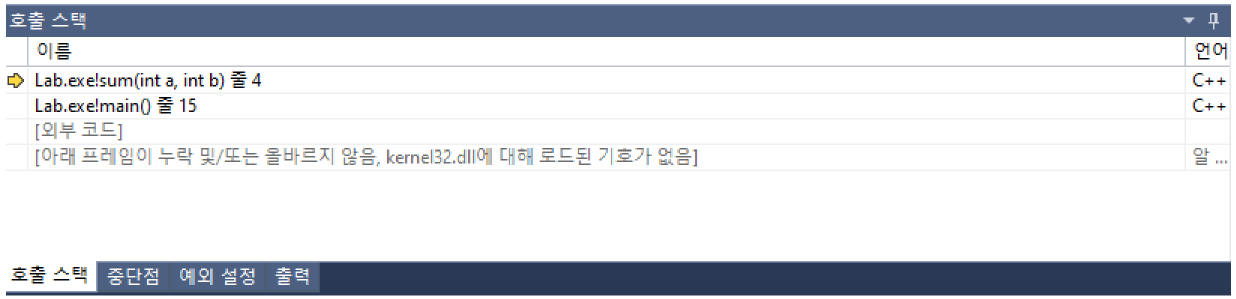
Other information



- 호출 스택: show stack frame of cuntions



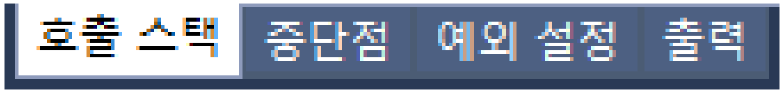
```
 1  #include <stdio.h>
 2
 3  int sum(int a, int b)
 4  {  경과 시간 1ms 이하
 5      int c = 0;
 6      c = a + b;
 7      return c;
 8  }
 9
10  int main()
11  {
12
13      int a = 10;
14      int b = 20;
15      int s = sum(a, b);
16      printf("sum = %d\n", s);
17
18      return 1;
19  }
```



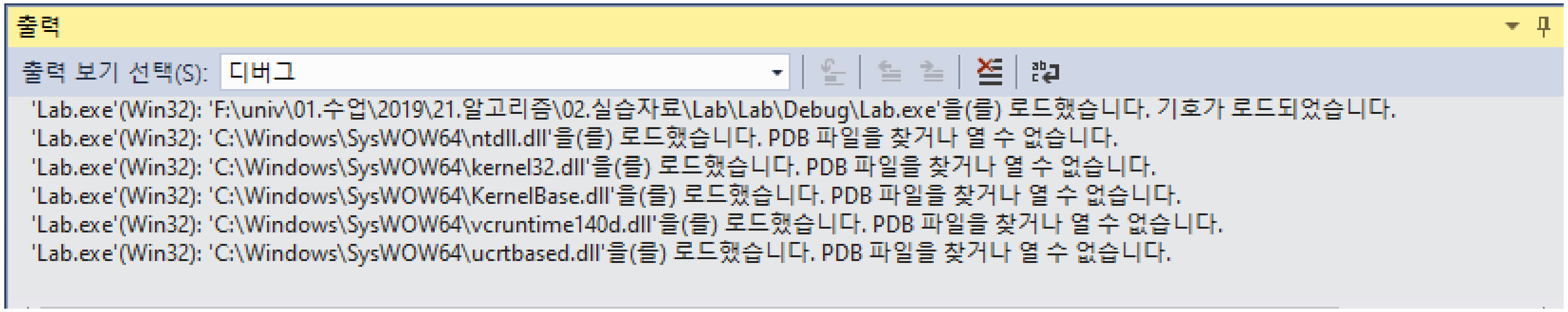
호출 스택

	이름	언어
➪	Lab.exe!sum(int a, int b) 줄 4	C++
	Lab.exe!main() 줄 15	C++
	[외부 코드]	
	[아래 프레임이 누락 및/또는 올바르지 않음, kernel32.dll에 대해 로드된 기호가 없음]	알 ...

Other information



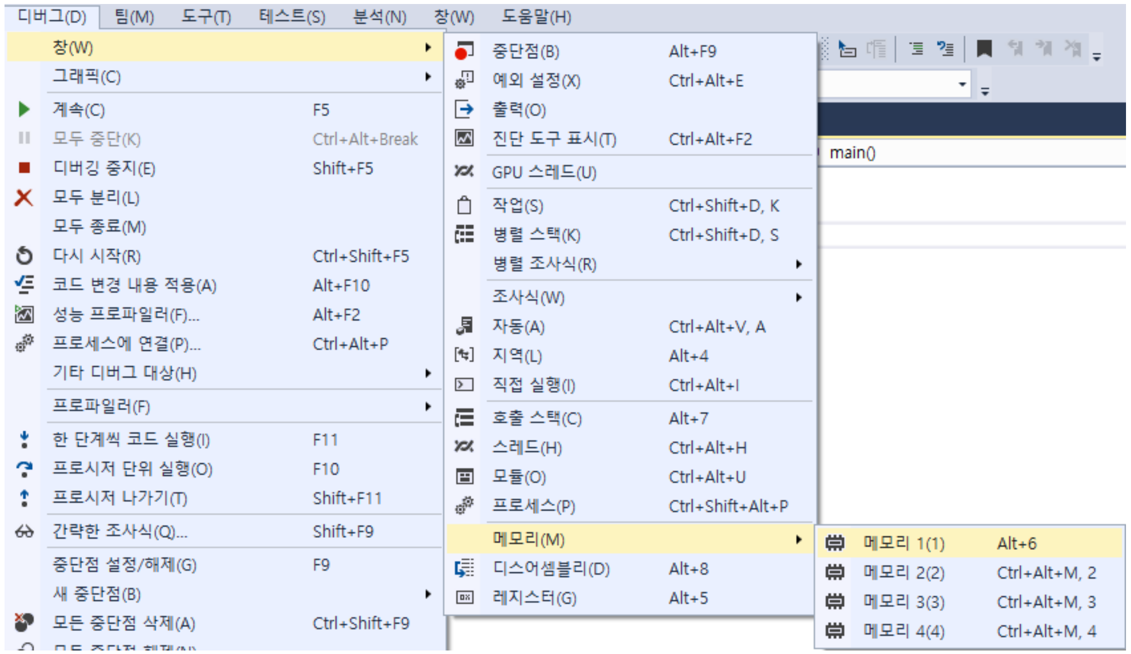
- 출력: Message output for debugging mode



```
'Lab.exe'(Win32): 'F:\univ\01.수업\2019\21.알고리즘\02.실습자료\Lab\Lab\Debug\Lab.exe'을(를) 로드했습니다. 기호가 로드되었습니다.
'Lab.exe'(Win32): 'C:\Windows\SysWOW64\ntdll.dll'을(를) 로드했습니다. PDB 파일을 찾거나 열 수 없습니다.
'Lab.exe'(Win32): 'C:\Windows\SysWOW64\kernel32.dll'을(를) 로드했습니다. PDB 파일을 찾거나 열 수 없습니다.
'Lab.exe'(Win32): 'C:\Windows\SysWOW64\KernelBase.dll'을(를) 로드했습니다. PDB 파일을 찾거나 열 수 없습니다.
'Lab.exe'(Win32): 'C:\Windows\SysWOW64\vcruntime140d.dll'을(를) 로드했습니다. PDB 파일을 찾거나 열 수 없습니다.
'Lab.exe'(Win32): 'C:\Windows\SysWOW64\ucrtbased.dll'을(를) 로드했습니다. PDB 파일을 찾거나 열 수 없습니다.
```

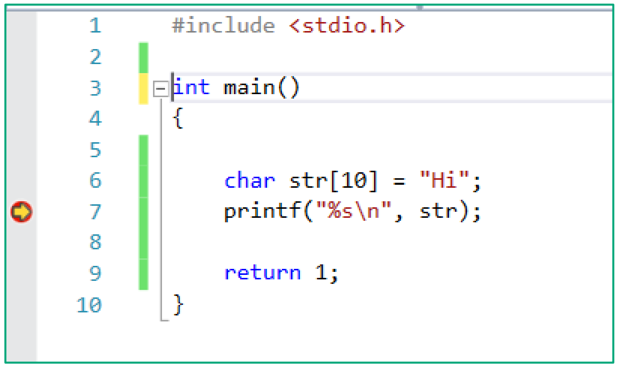
Other information

- Memory usage

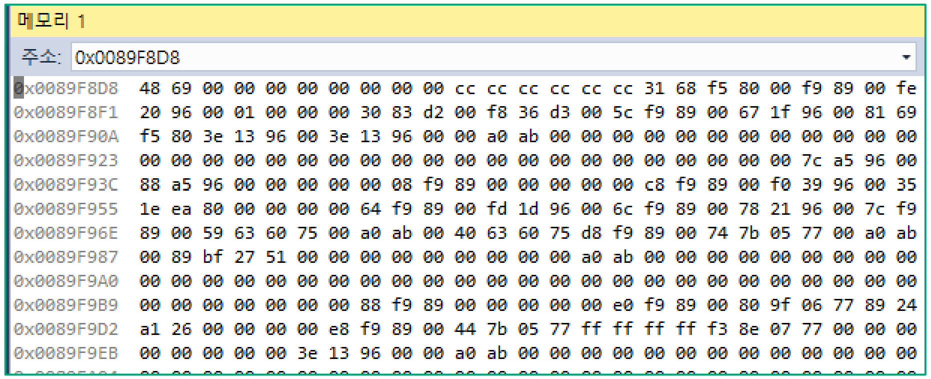


Other information

- Memory usage



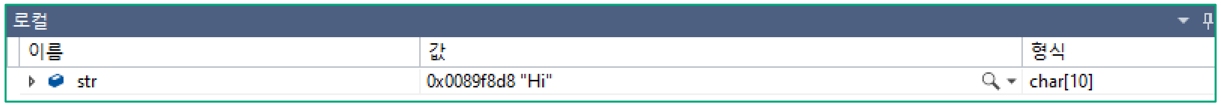
```c
#include <stdio.h>

int main()
{

    char str[10] = "Hi";
    printf("%s\n", str);

    return 1;
}
```



메모리 1

주소: 0x0089F8D8

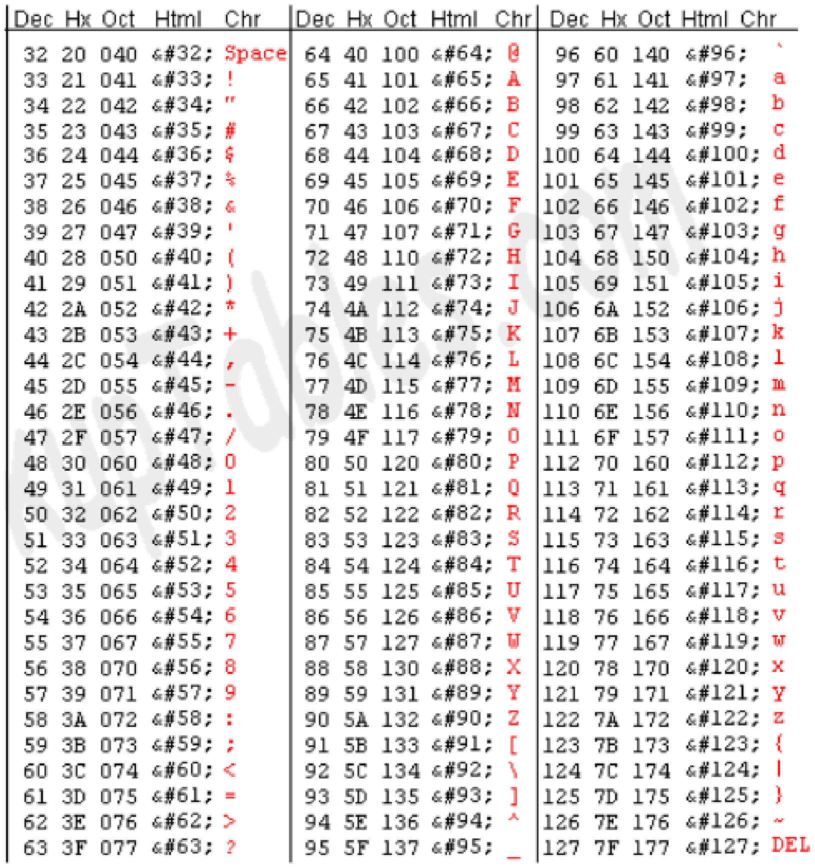
```
0x0089F8D8  48 69 00 00 00 00 00 00 00 00 cc cc cc cc cc cc 31 68 f5 80 00 f9 89 00 fe
0x0089F8F1  20 96 00 01 00 00 00 30 83 d2 00 f8 36 d3 00 5c f9 89 00 67 1f 96 00 81 69
0x0089F90A  f5 80 3e 13 96 00 3e 13 96 00 00 a0 ab 00 00 00 00 00 00 00 00 00 00 00 00
0x0089F923  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7c a5 96 00
0x0089F93C  88 a5 96 00 00 00 00 00 08 f9 89 00 00 00 00 00 c8 f9 89 00 f0 39 96 00 35
0x0089F955  1e ea 80 00 00 00 00 64 f9 89 00 fd 1d 96 00 6c f9 89 00 78 21 96 00 7c f9
0x0089F96E  89 00 59 63 60 75 00 a0 ab 00 40 63 60 75 d8 f9 89 00 74 7b 05 77 00 a0 ab
0x0089F987  00 89 bf 27 51 00 00 00 00 00 00 00 00 00 a0 ab 00 00 00 00 00 00 00 00 00
0x0089F9A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0089F9B9  00 00 00 00 00 00 00 88 f9 89 00 00 00 00 00 e0 f9 89 00 80 9f 06 77 89 24
0x0089F9D2  a1 26 00 00 00 00 e8 f9 89 00 44 7b 05 77 ff ff ff ff f3 8e 07 77 00 00 00
0x0089F9EB  00 00 00 00 00 3e 13 96 00 00 a0 ab 00 00 00 00 00 00 00 00 00 00 00 00 00
```



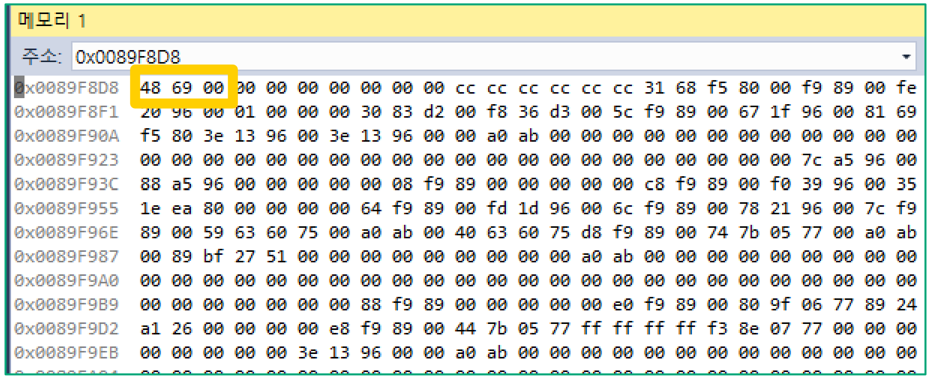
로컬

이름	값	형식
str	0x0089f8d8 "Hi"	char[10]

Other information



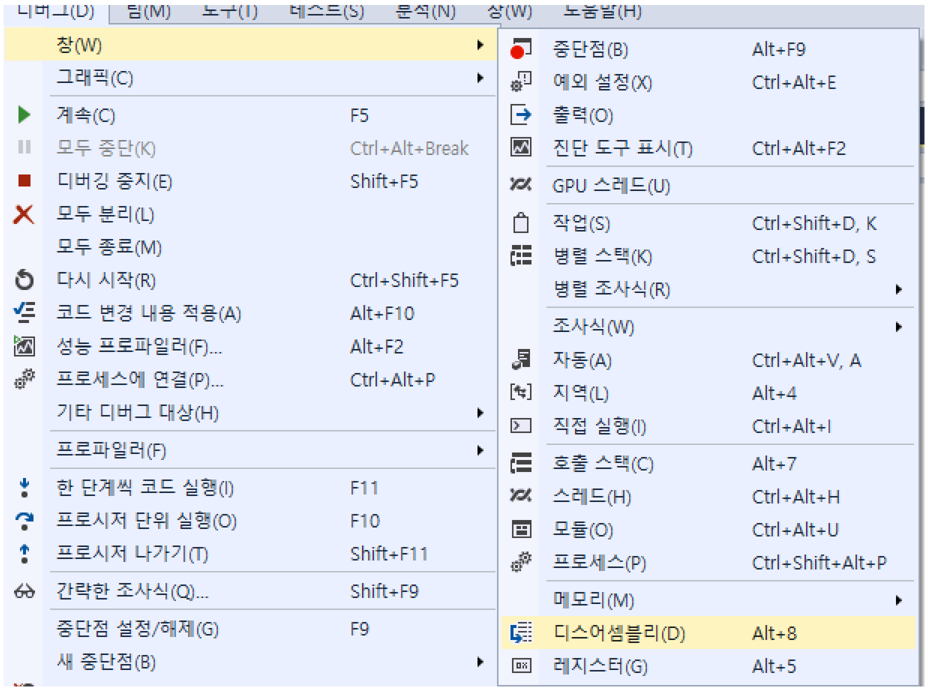
Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL



```
메모리 1
주소: 0x0089F8D8
0x0089F8D8  48 69 00 00 00 00 00 00 00 00 cc cc cc cc cc cc 31 68 f5 80 00 f9 89 00 fe
0x0089F8F1  20 96 00 01 00 00 00 30 83 d2 00 f8 36 d3 00 5c f9 89 00 67 1f 96 00 81 69
0x0089F90A  f5 80 3e 13 96 00 3e 13 96 00 00 a0 ab 00 00 00 00 00 00 00 00 00 00 00 00
0x0089F923  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7c a5 96 00
0x0089F93C  88 a5 96 00 00 00 00 00 08 f9 89 00 00 00 00 00 c8 f9 89 00 f0 39 96 00 35
0x0089F955  1e ea 80 00 00 00 00 64 f9 89 00 fd 1d 96 00 6c f9 89 00 78 21 96 00 7c f9
0x0089F96E  89 00 59 63 60 75 00 a0 ab 00 40 63 60 75 d8 f9 89 00 74 7b 05 77 00 a0 ab
0x0089F987  00 89 bf 27 51 00 00 00 00 00 00 00 00 00 a0 ab 00 00 00 00 00 00 00 00 00
0x0089F9A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0089F9B9  00 00 00 00 00 00 00 88 f9 89 00 00 00 00 00 e0 f9 89 00 80 9f 06 77 89 24
0x0089F9D2  a1 26 00 00 00 00 e8 f9 89 00 44 7b 05 77 ff ff ff ff f3 8e 07 77 00 00 00
0x0089F9EB  00 00 00 00 00 3e 13 96 00 00 a0 ab 00 00 00 00 00 00 00 00 00 00 00 00 00
```

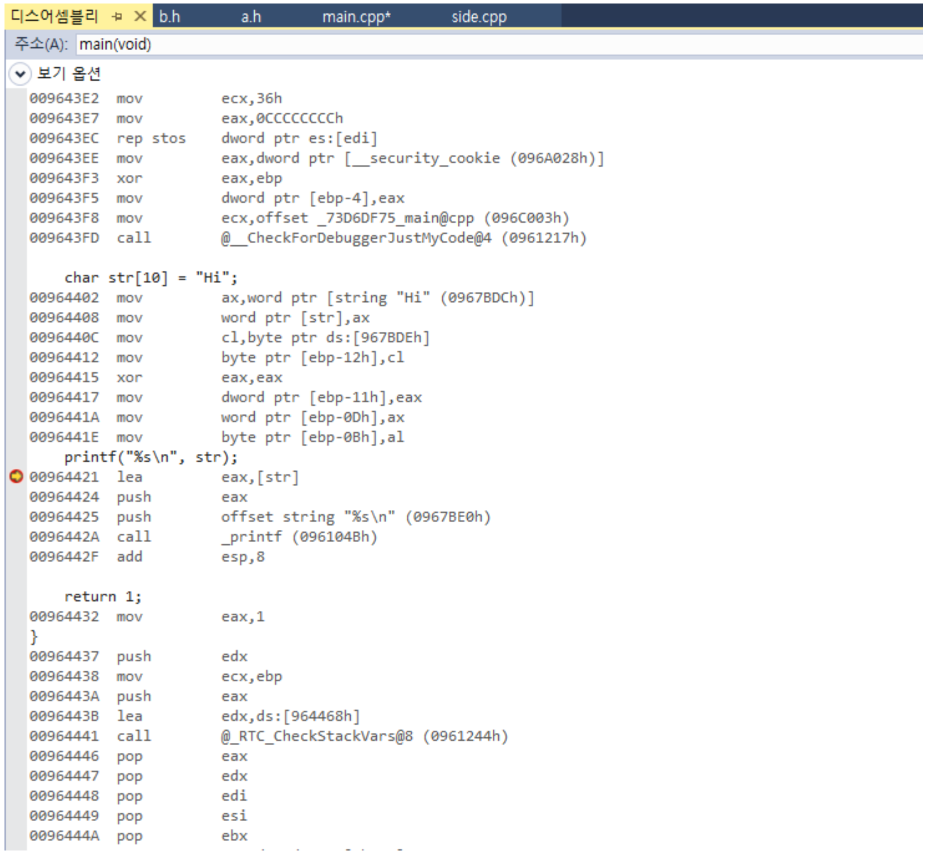
Other information

- disassembly



Other information

- disassembly



```
009643E2  mov         ecx,36h
009643E7  mov         eax,0CCCCCCCCh
009643EC  rep stos    dword ptr es:[edi]
009643EE  mov         eax,dword ptr [__security_cookie (096A028h)]
009643F3  xor         eax,ebp
009643F5  mov         dword ptr [ebp-4],eax
009643F8  mov         ecx,offset _73D6DF75_main@cpp (096C003h)
009643FD  call        @__CheckForDebuggerJustMyCode@4 (0961217h)

    char str[10] = "Hi";
00964402  mov         ax,word ptr [string "Hi" (0967BDCh)]
00964408  mov         word ptr [str],ax
0096440C  mov         cl,byte ptr ds:[967BDEh]
00964412  mov         byte ptr [ebp-12h],cl
00964415  xor         eax,eax
00964417  mov         dword ptr [ebp-11h],eax
0096441A  mov         word ptr [ebp-0Dh],ax
0096441E  mov         byte ptr [ebp-0Bh],al
    printf("%s\n", str);
00964421  lea         eax,[str]
00964424  push        eax
00964425  push        offset string "%s\n" (0967BE0h)
0096442A  call        _printf (096104Bh)
0096442F  add         esp,8

    return 1;
00964432  mov         eax,1
}
00964437  push        edx
00964438  mov         ecx,ebp
0096443A  push        eax
0096443B  lea         edx,ds:[964468h]
00964441  call        @_RTC_CheckStackVars@8 (0961244h)
00964446  pop         eax
00964447  pop         edx
00964448  pop         edi
00964449  pop         esi
0096444A  pop         ebx
```

End of Lecture