Memory structure

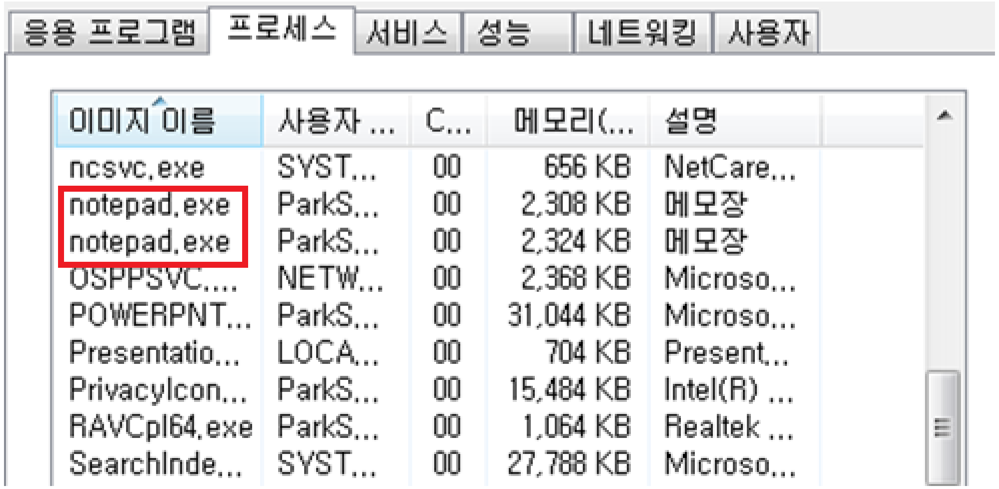
Sanghoon, Lee

Multidimensional
insight

Terminology

- **Program : What you want to do using computer**
- **Process : The instance of a program that is being executed and allocated on memory**
- **Thread : A component of a process.**

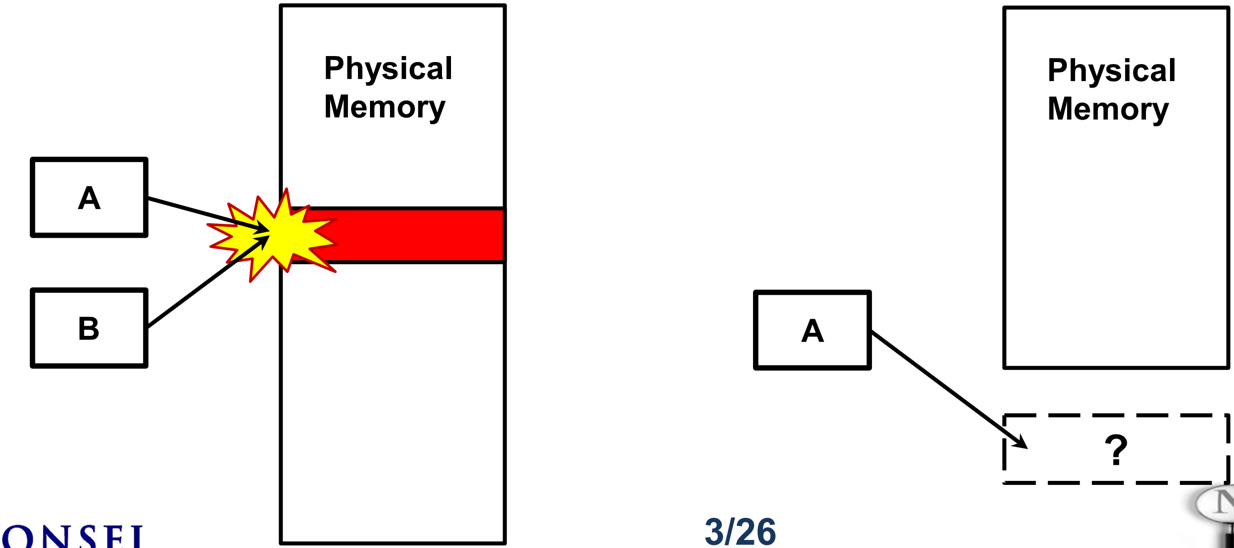
➢ **When you run two ‘note pad’, one same program and different two processes will be activated**



이미지 이름	사용자 ...	C...	메모리(...	설명
ncsvc.exe	SYST...	00	656 KB	NetCare...
notepad.exe	ParkS...	00	2,308 KB	메모장
notepad.exe	ParkS...	00	2,324 KB	메모장
OSPPSVC....	NETW...	00	2,368 KB	Microso...
POWERPNT...	ParkS...	00	31,044 KB	Microso...
Presentatio...	LOCA...	00	704 KB	Present...
PrivacyIcon...	ParkS...	00	15,484 KB	Intel(R) ...
RAVCpl64.exe	ParkS...	00	1,064 KB	Realtek ...
SearchInde...	SYST...	00	27,788 KB	Microso...

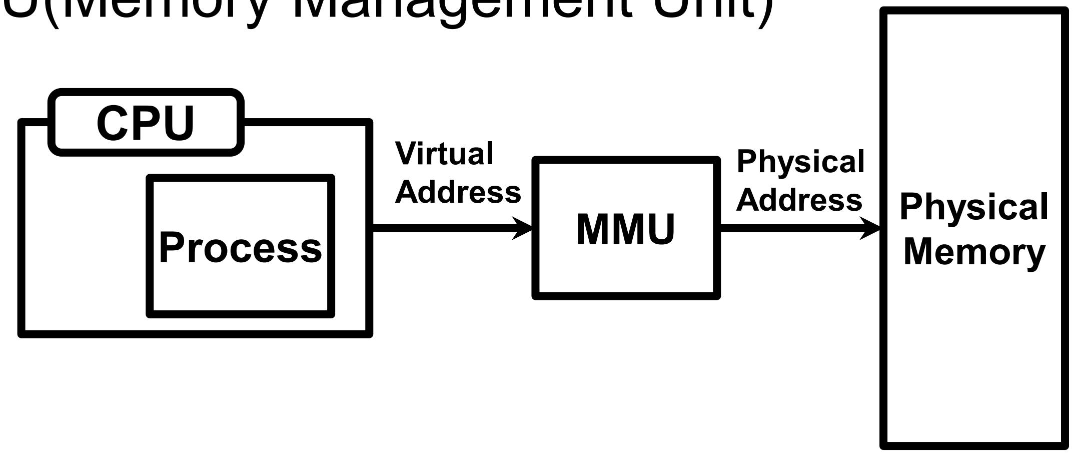
Physical memory

- **Mainly, physical memory indicates RAM(primary memory)**
- **Problems :**
  - What if processes A and B want to use a same physical address (or physical memory is already fully occupied)
  - What if a demanded address does not exist, because of too small RAM memory size



Virtual memory

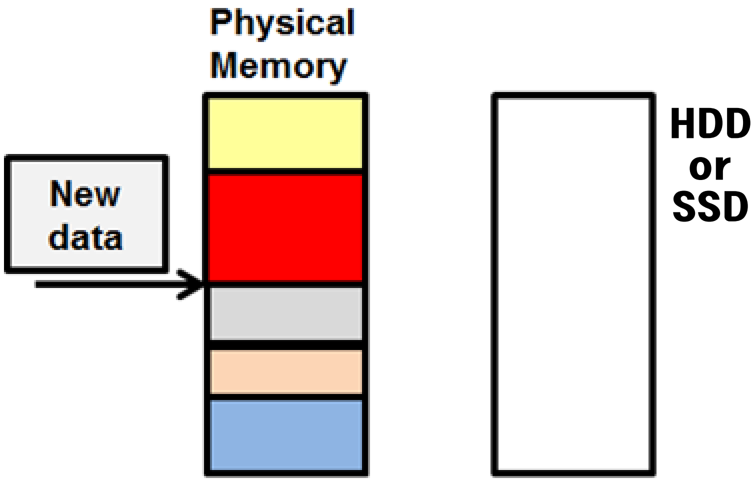
- **What is a virtual memory?**
  - Proposed to manage a physical memory efficiently
  - Each process has its own virtual memory (4Gb for all 32bits OS)
  - Each process can use the whole virtual memory whether the physical memory is less than 4Gb or not
  - Virtual memory is translated to physical memory by MMU(Memory Management Unit)



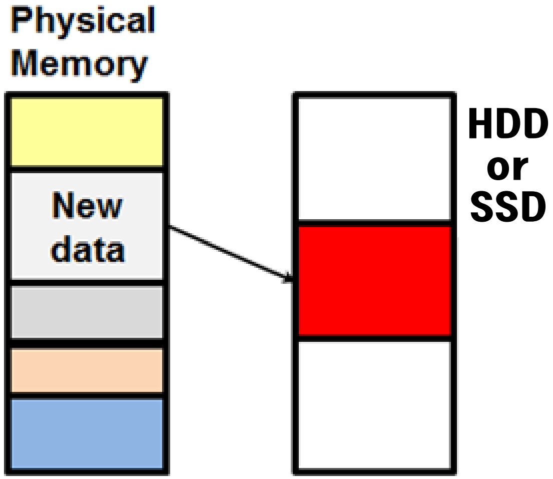
Virtual memory

- **What is a virtual memory?(Continued)**
  - MMU deceives CPU as like it uses a continuous physical memory while using a discrete physical memory
  - Using the discontinuous physical memory space, MMU can deceive CPU as if CPU uses continuous physical memory
  - If there is not enough RAM memory, it can use a part of HDD(or SSD) as a **Physical memory.** So you can use 4Gb or more memory with small RAM memory

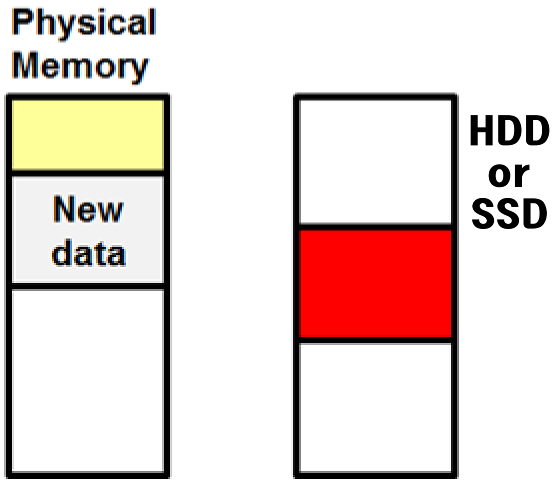
Virtual memory



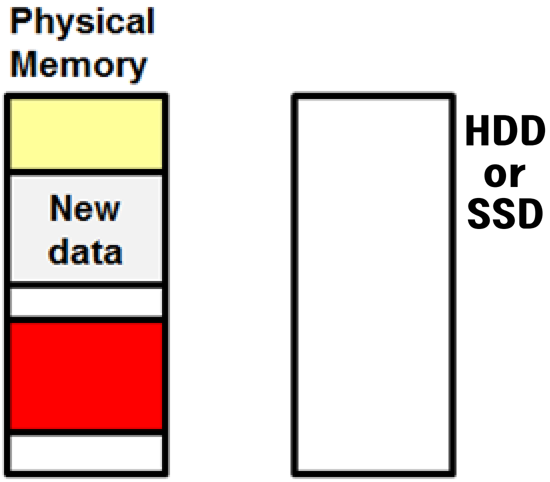
1. When new data gets into fully occupied RAM,



2. ‘Unused data’ goes to HDD and new data takes the RAM space



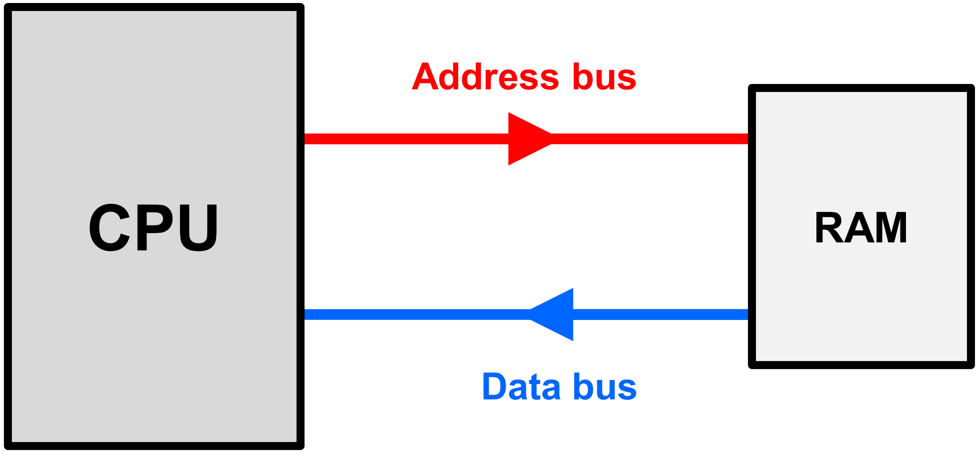
3. When some memories are released



4. ‘Unused data’ can go back into the RAM space, if needed

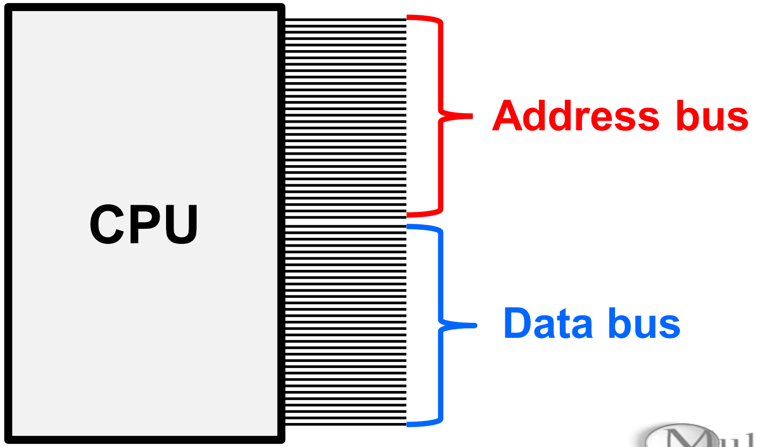
Virtual memory

- **What is a 32 bit computer?**
  - A sequence how CPU gets data from RAM
    1. CPU sends an address to RAM through the address bus
    2. RAM finds data using the received address from CPU
    3. Finally, RAM sends the requested data to CPU through the data bus



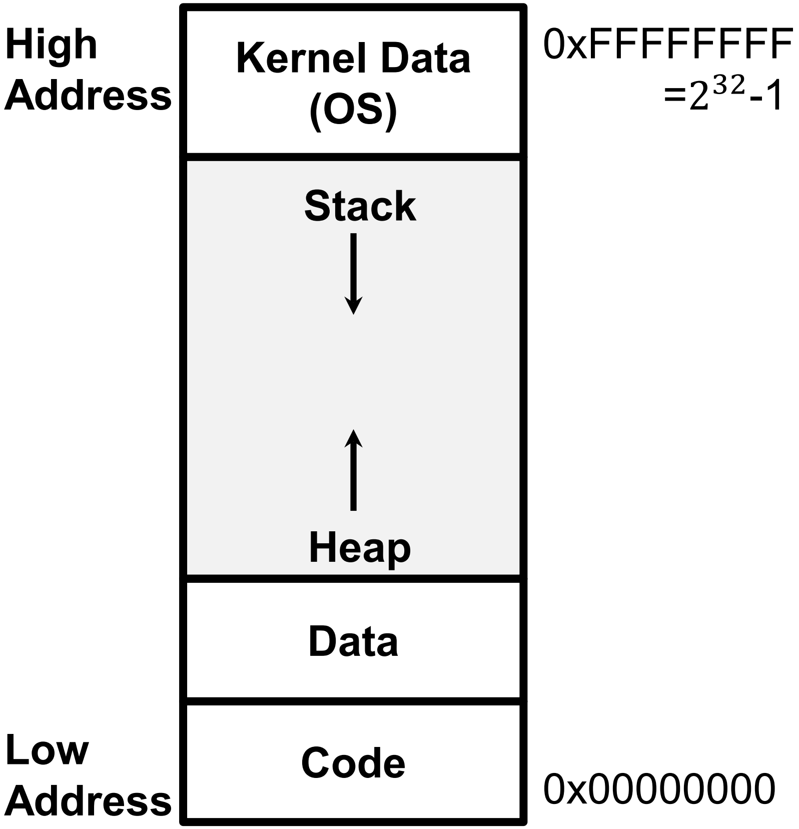
Virtual memory

- In a 32 bit computer, both address and data bus have 32 pins
- Each pin can represent 1bit (0 or 1). So, 32 pins can represent 32 bit data and address
- RAM can send 32bit data to CPU for each clock cycle
- 32 bit can represent $2^{32} = 4G$ cases. So ideally, 32bit computer can use 4Gb RAM in maximum.
  - It can be a reason for why virtual memory is designed to be 4Gb memory

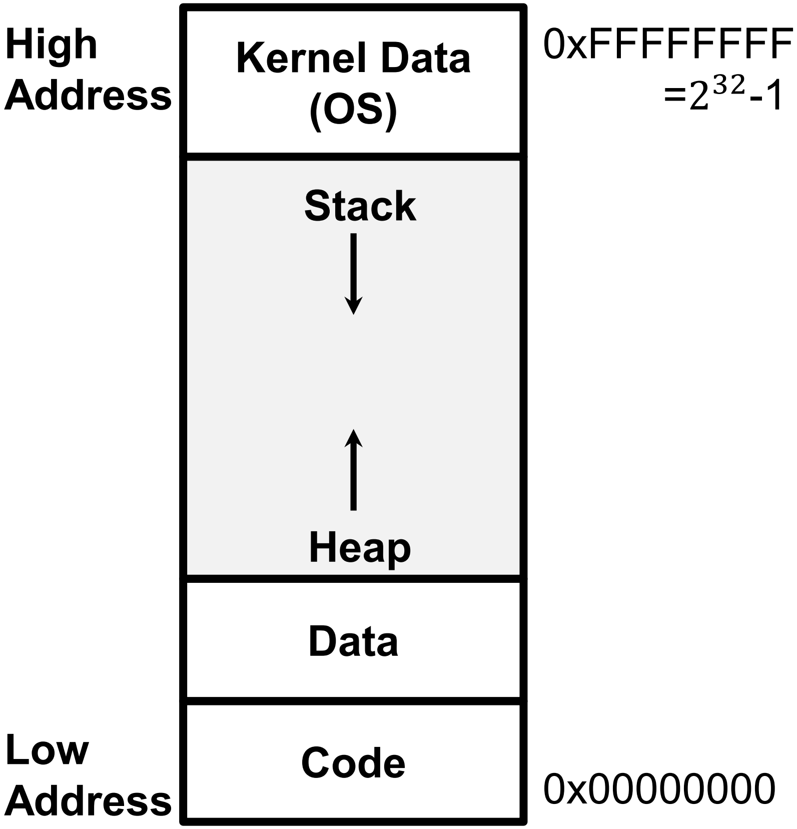


Virtual memory

- **The structure of virtual memory (32bit computer) (≈ structure of process loaded on memory)**



Virtual memory



- **Kernel Data region**

  : Region for kernel(≈ OS)

- **Stack**

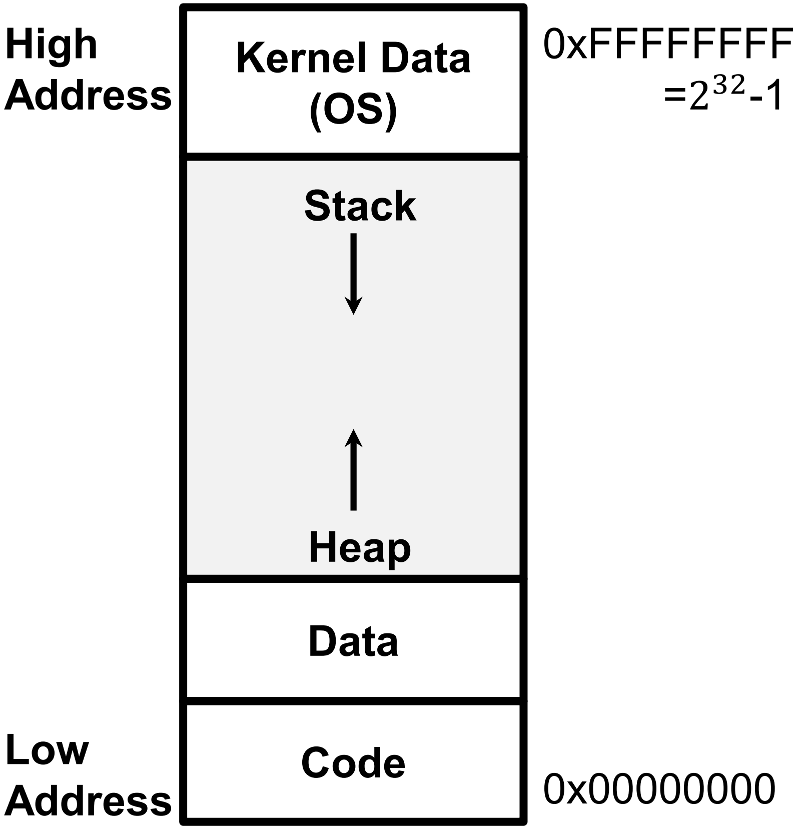
  : Based on LIFO(Last In First Out) structure

- **Heap**

  : almost same as a priority queue. User can control allocation and release timing

➢ We will cover stack and heap in another lecture

Virtual memory



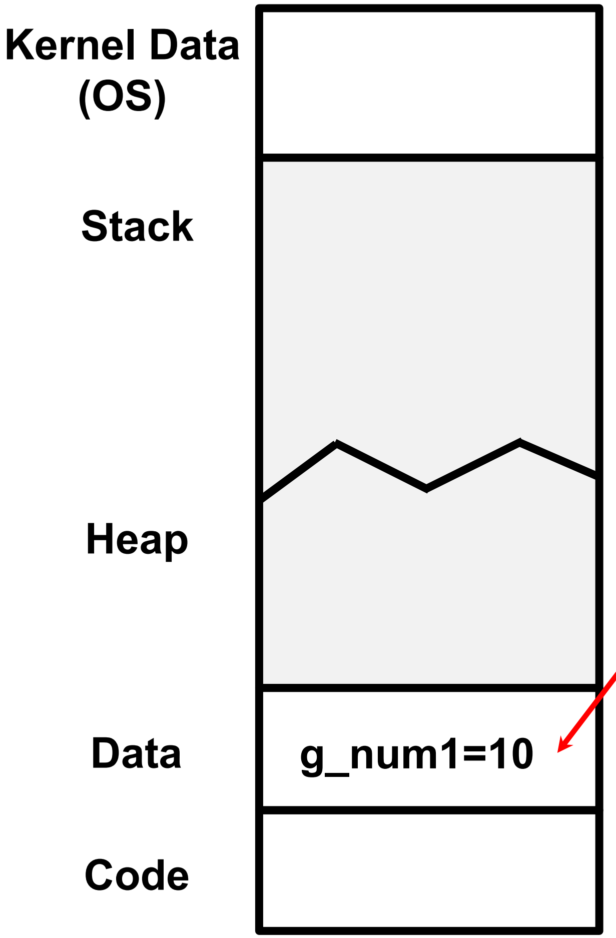
- **Data**

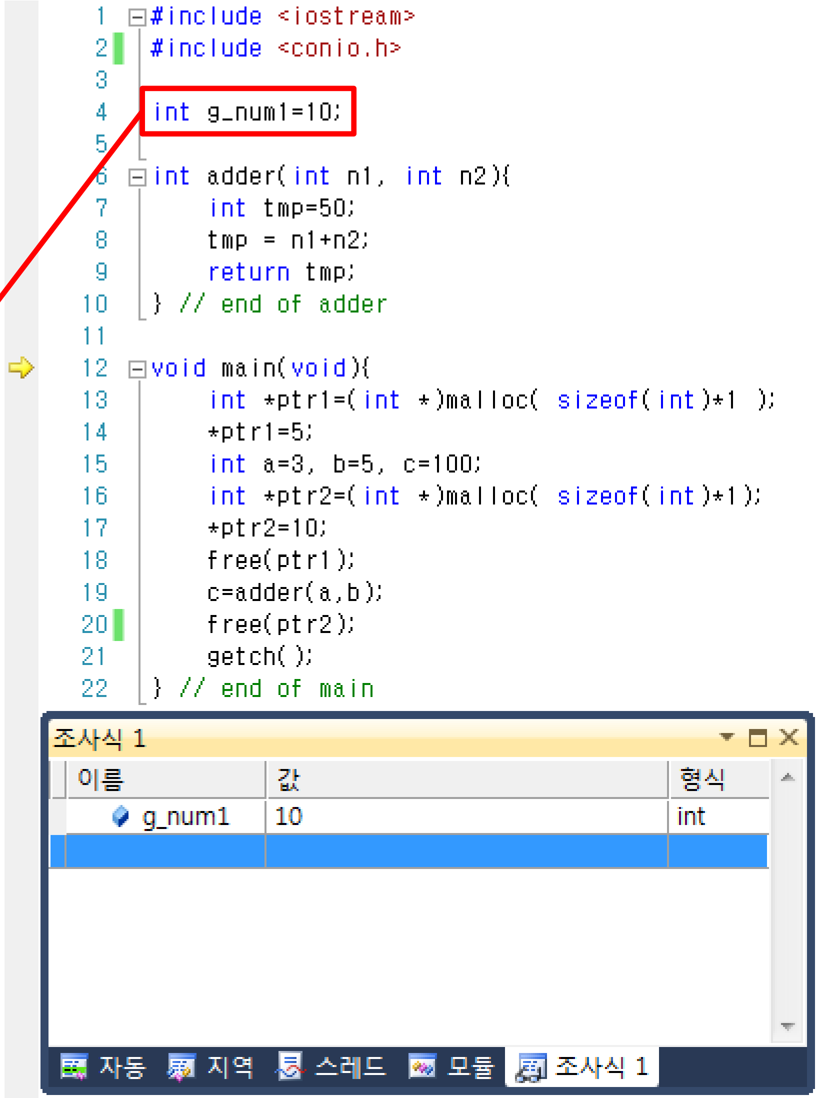
  : Region for static and global variables

- **Code**

  : Region for ‘C language’ code. Strictly saying, it is a region for binary version of code.

Virtual memory(1/15)



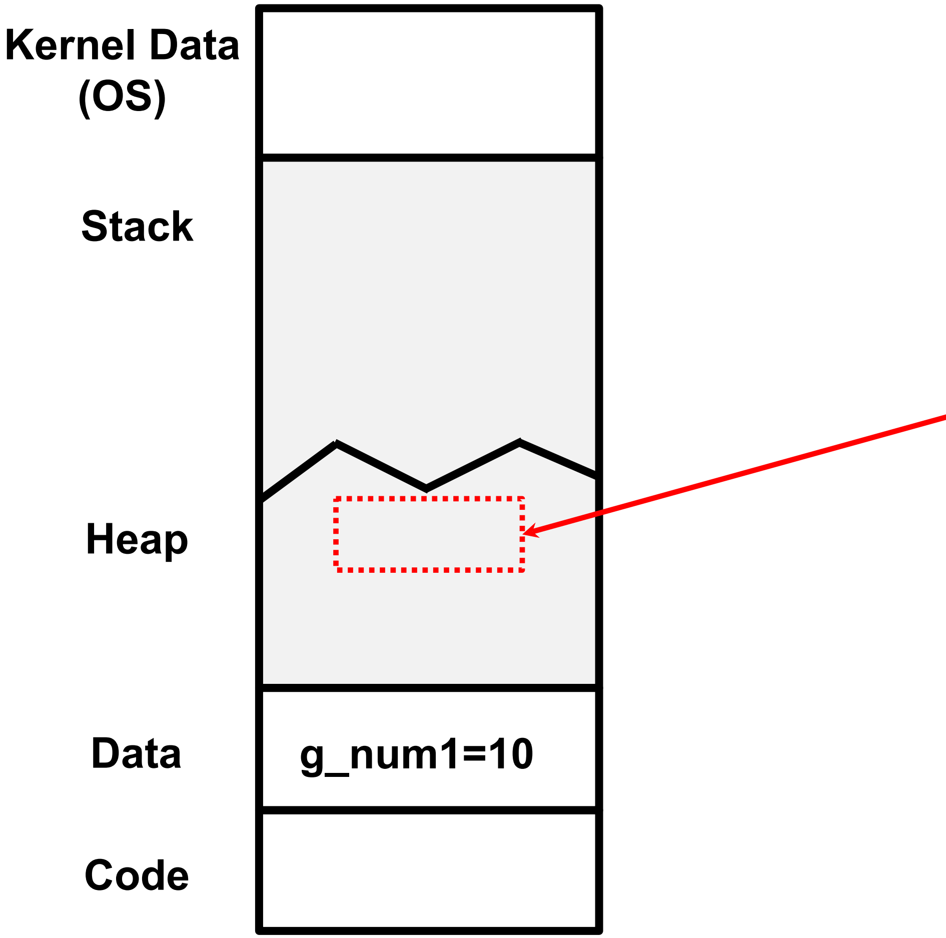


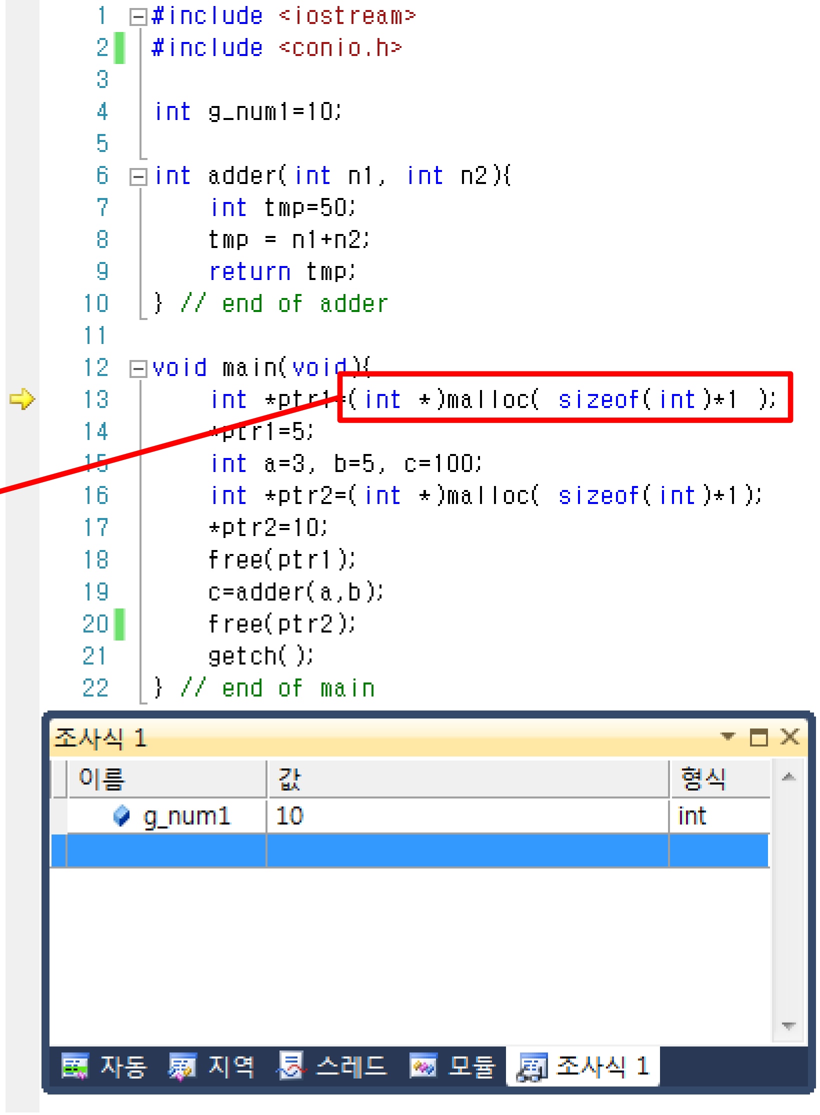
```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder( int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=(int *)malloc( sizeof(int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=(int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch();
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int

Virtual memory(2/15)



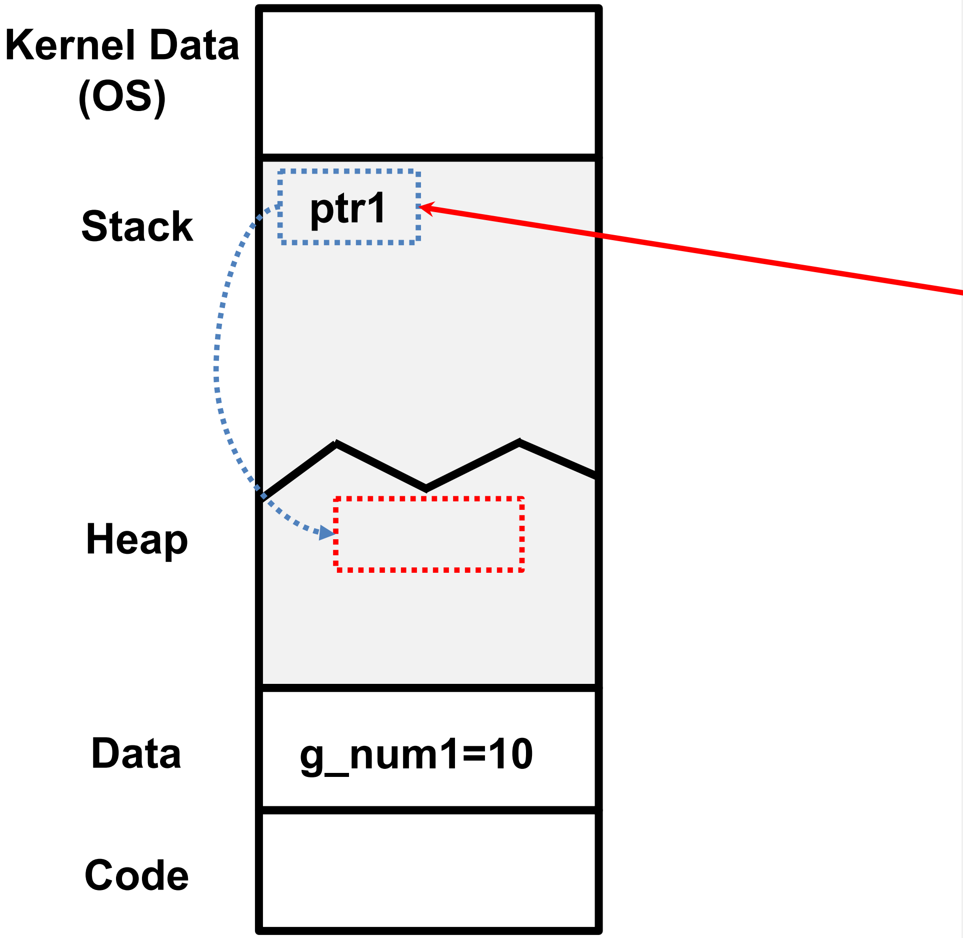


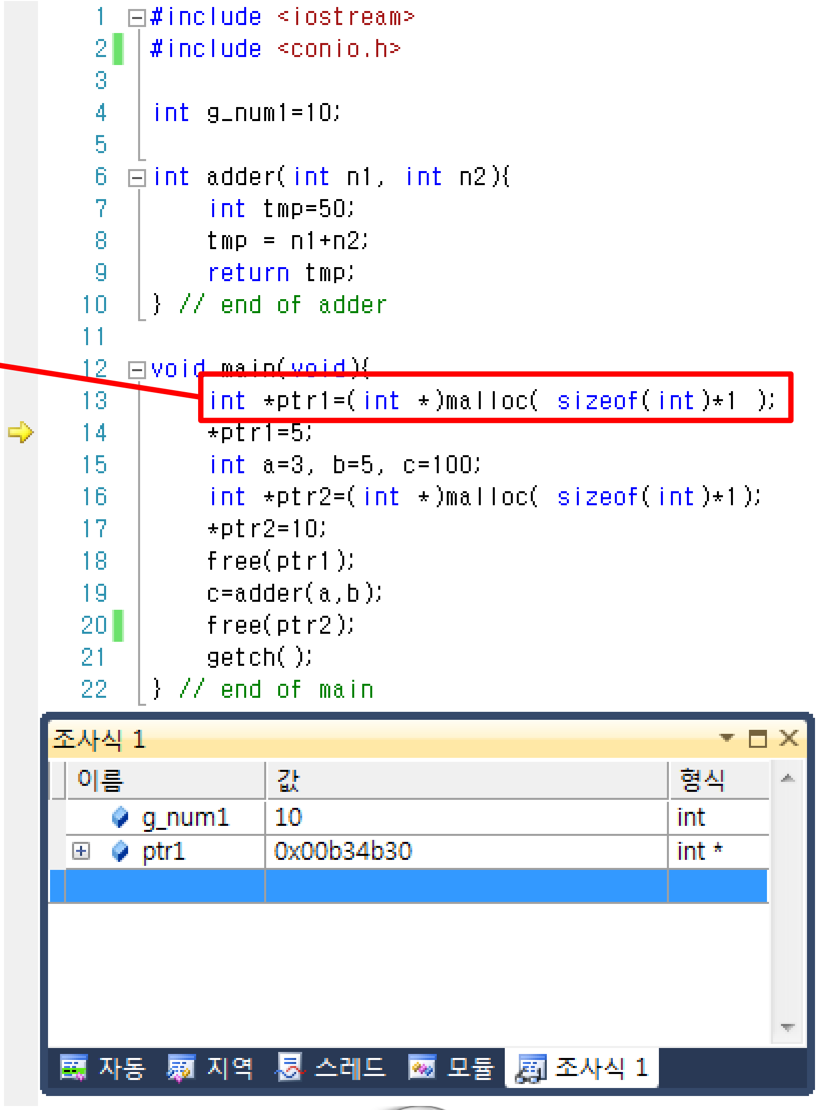
```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder(int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=(int *)malloc( sizeof(int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=(int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch();
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int

Virtual memory(3/15)



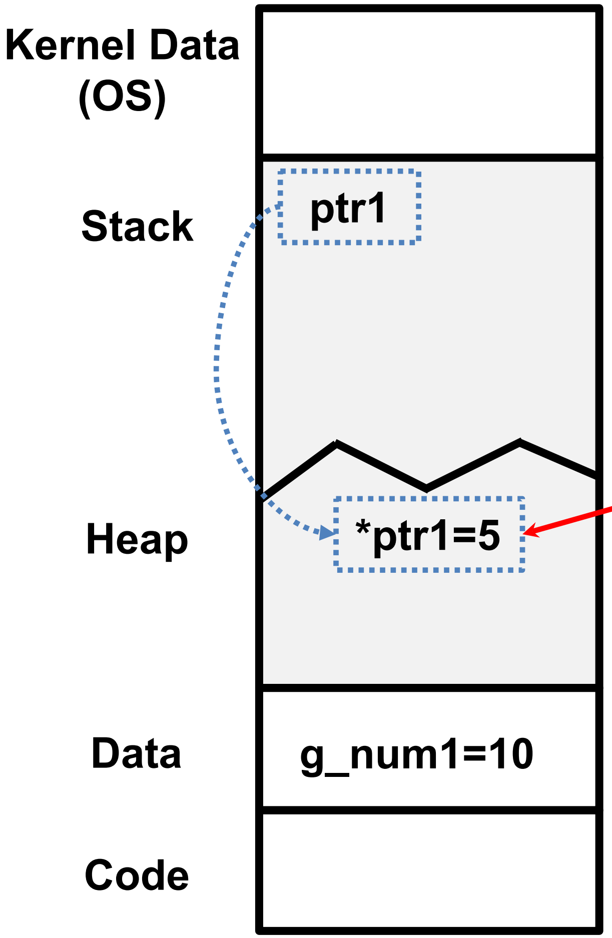


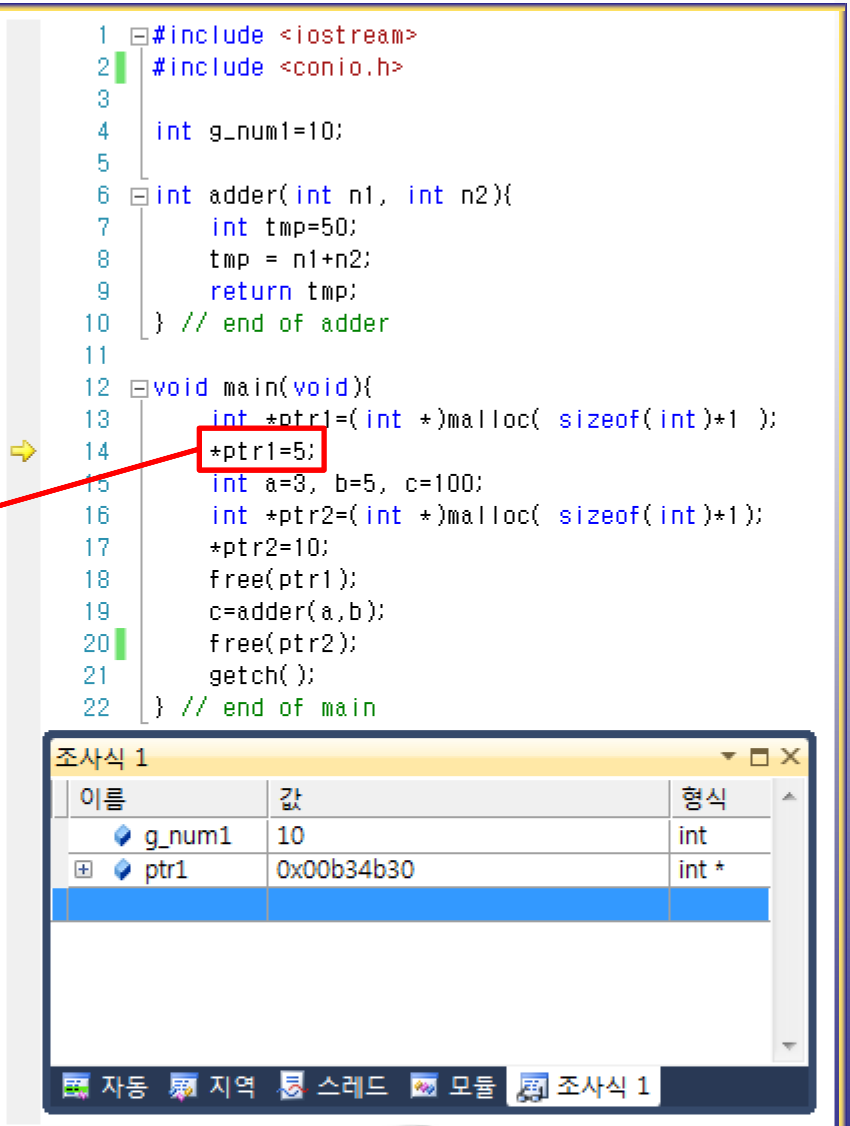
```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder(int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=(int *)malloc( sizeof(int)*1  );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=(int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch();
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int
⊞ ptr1	0x00b34b30	int *

Virtual memory(4/15)



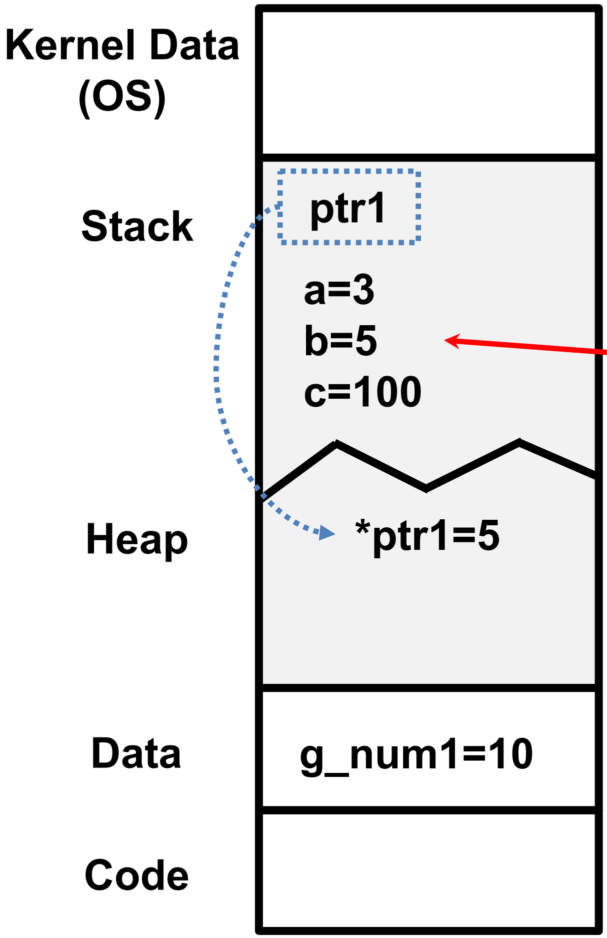


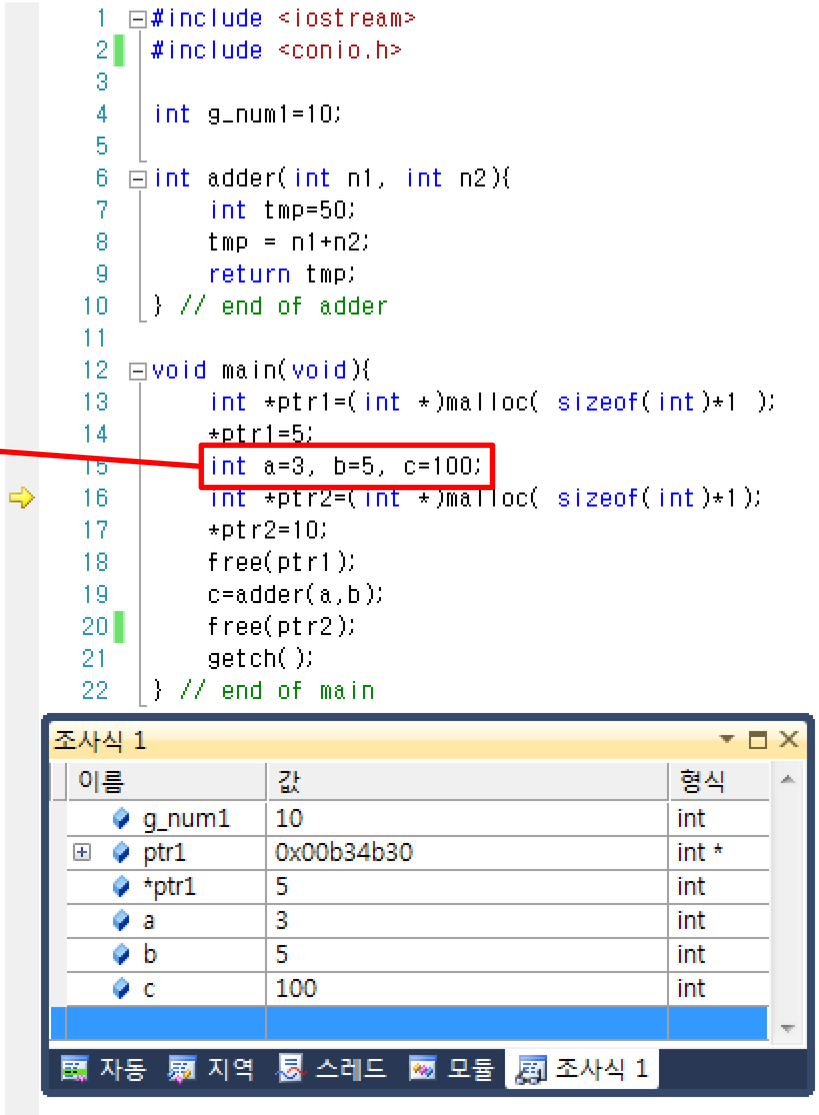
```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder( int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=( int *)malloc( sizeof( int )*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=( int *)malloc( sizeof( int )*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch( );
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int
ptr1	0x00b34b30	int *

Virtual memory(5/15)

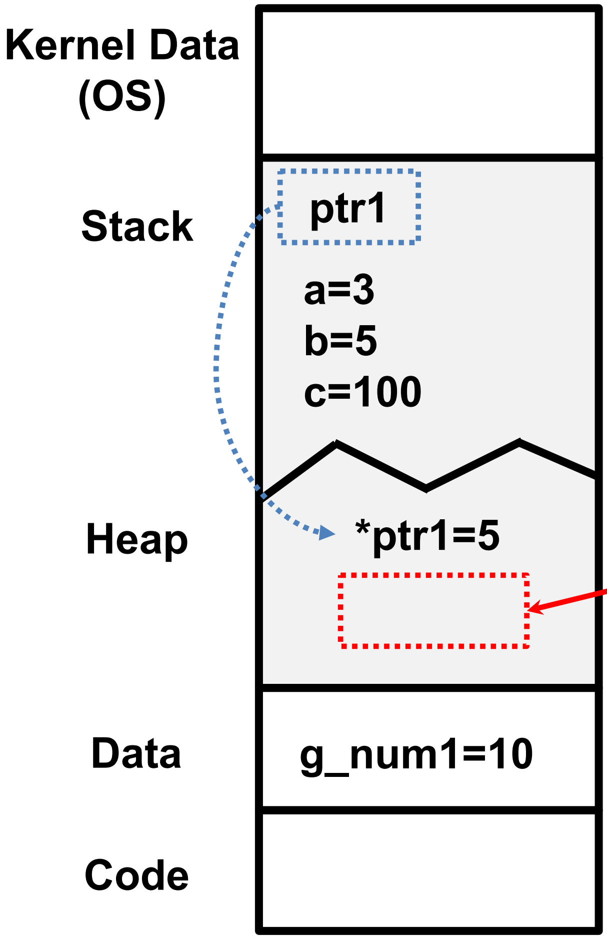




```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder( int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=( int *)malloc( sizeof(int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=( int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch( );
} // end of main
```

이름	값	형식
g_num1	10	int
ptr1	0x00b34b30	int *
*ptr1	5	int
a	3	int
b	5	int
c	100	int

Virtual memory(6/15)



Kernel Data (OS)

Stack

ptr1

a=3
b=5
c=100

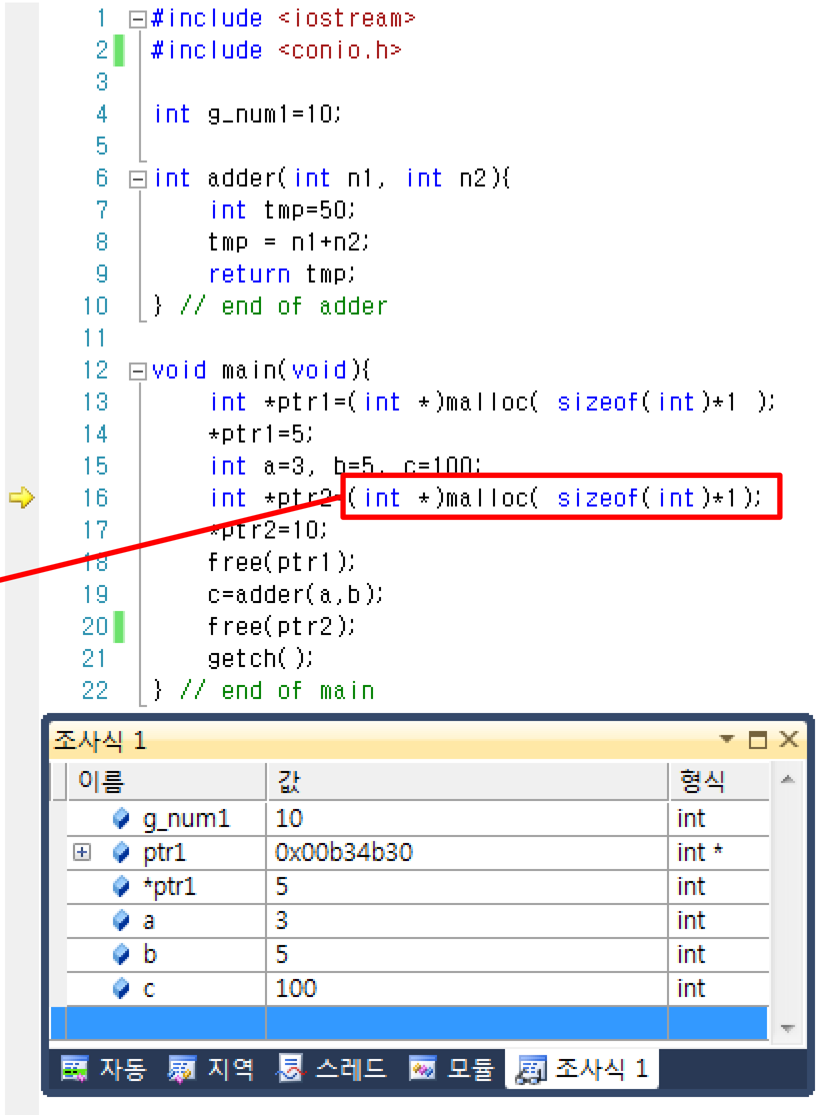
Heap

*ptr1=5

Data

g_num1=10

Code

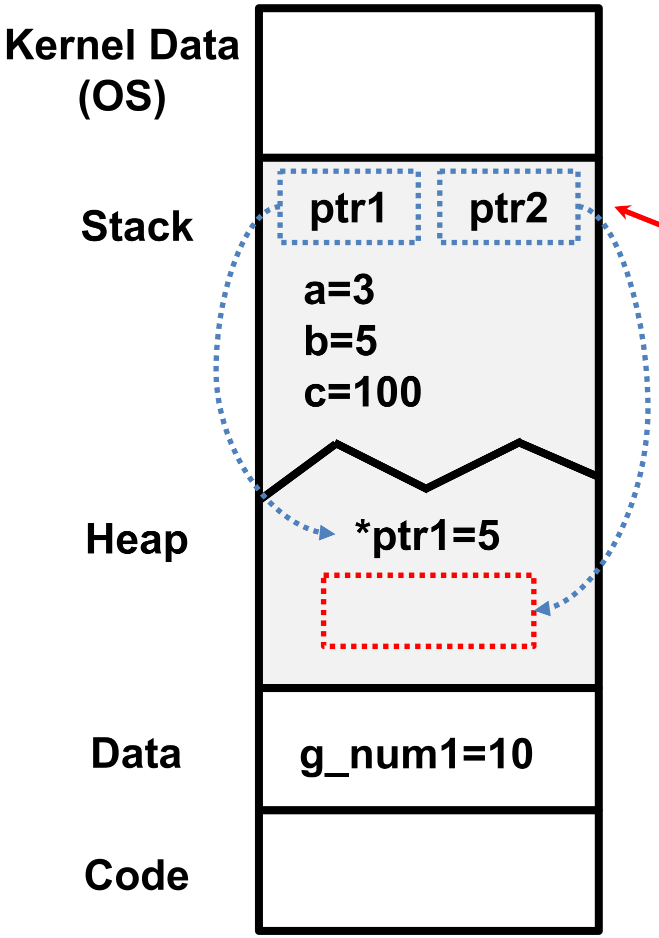


```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder( int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=(int *)malloc( sizeof(int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=(int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch();
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int
ptr1	0x00b34b30	int *
*ptr1	5	int
a	3	int
b	5	int
c	100	int

Virtual memory(7/15)



Kernel Data (OS)

Stack

ptr1

ptr2

a=3
b=5
c=100

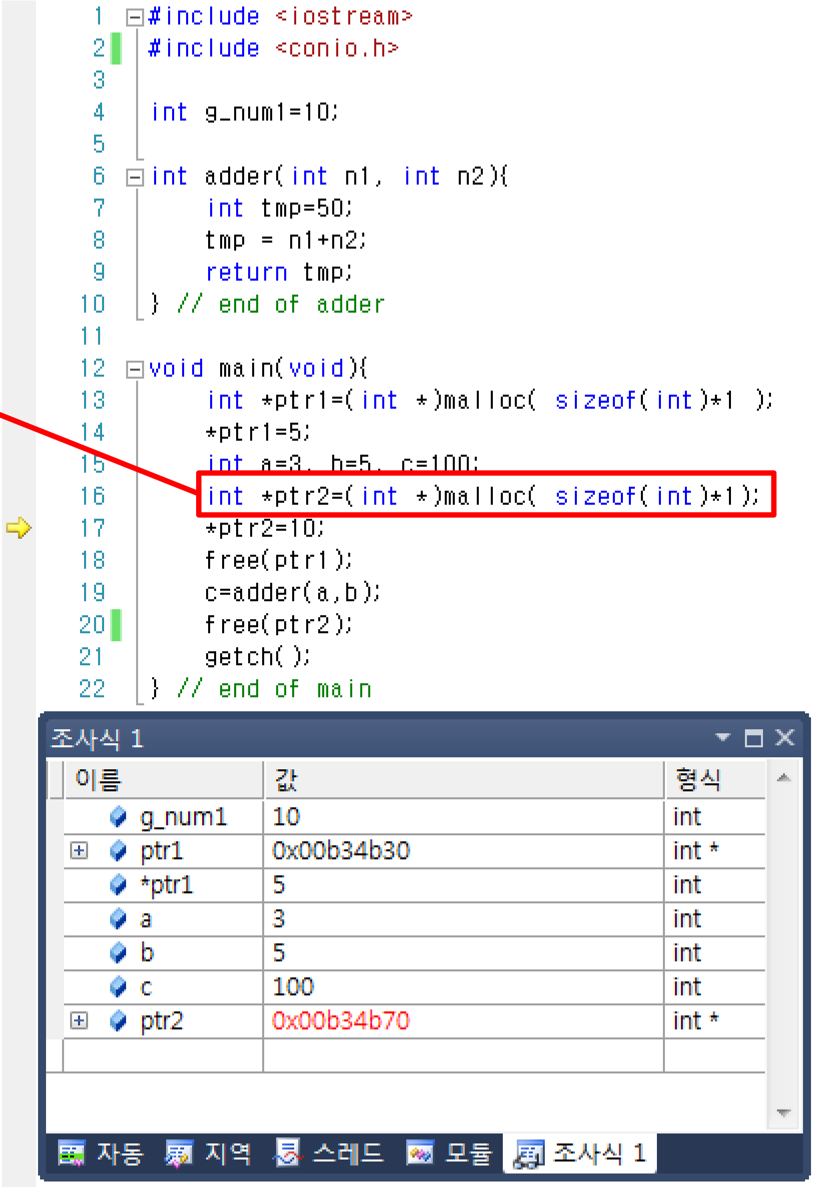
Heap

*ptr1=5

Data

g_num1=10

Code

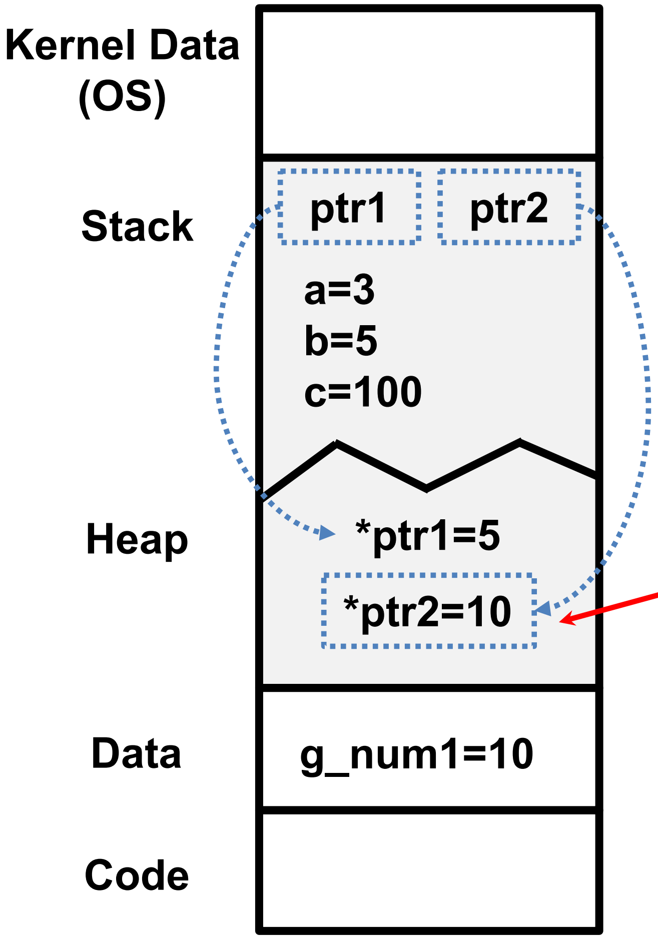


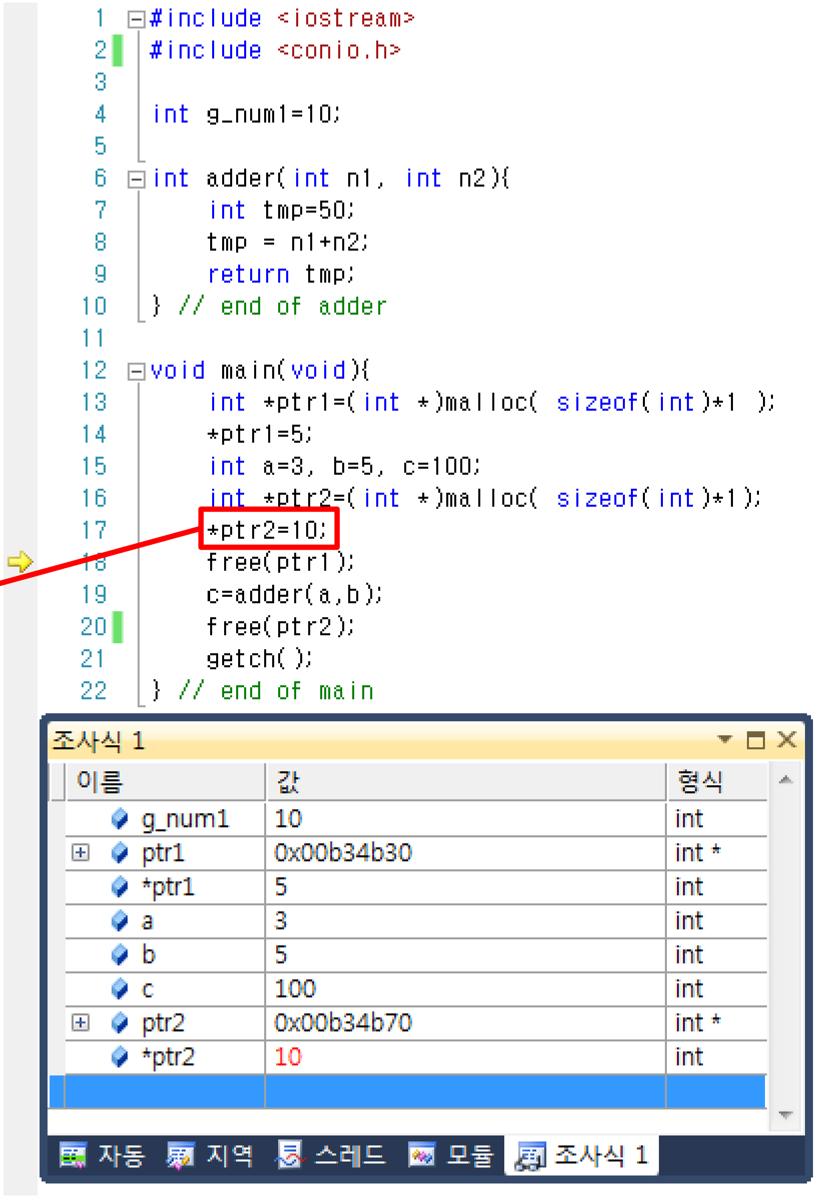
```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder( int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=( int *)malloc( sizeof( int )*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=( int *)malloc( sizeof( int )*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch( );
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int
⊞ ptr1	0x00b34b30	int *
*ptr1	5	int
a	3	int
b	5	int
c	100	int
⊞ ptr2	0x00b34b70	int *

Virtual memory(8/15)



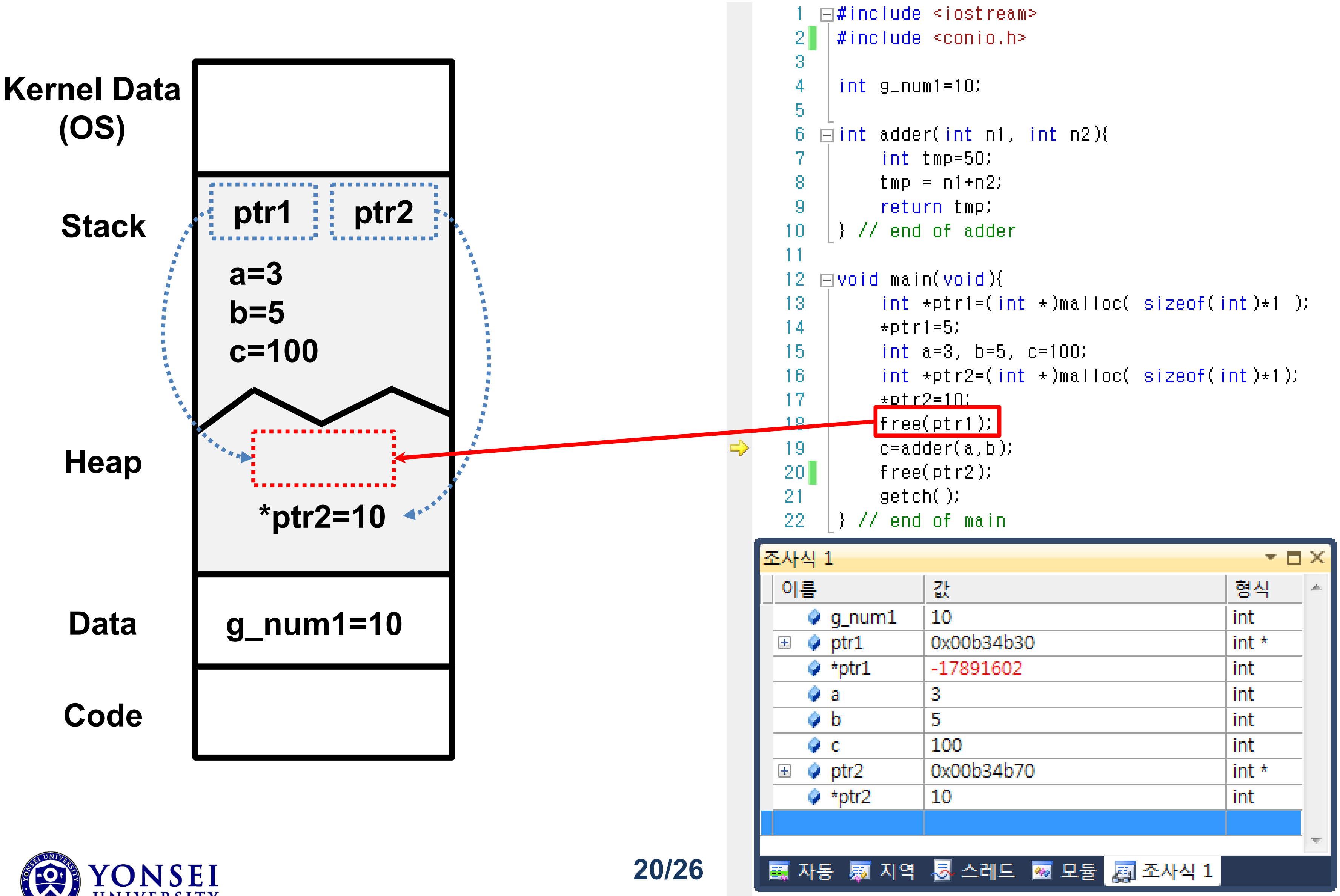


```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder(int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=(int *)malloc( sizeof(int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=(int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch();
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int
ptr1	0x00b34b30	int *
*ptr1	5	int
a	3	int
b	5	int
c	100	int
ptr2	0x00b34b70	int *
*ptr2	10	int

Virtual memory(9/15)

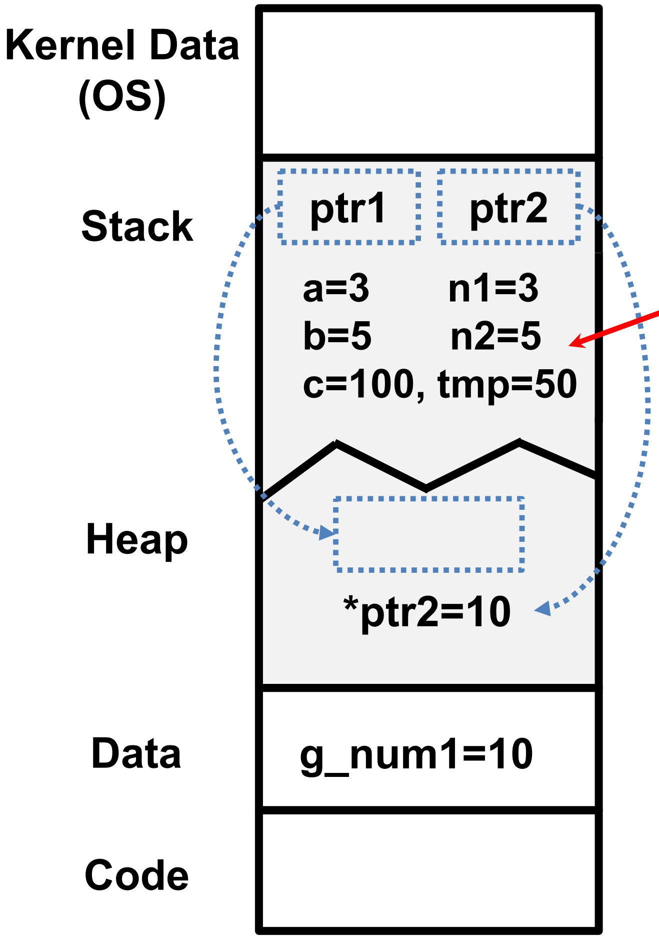


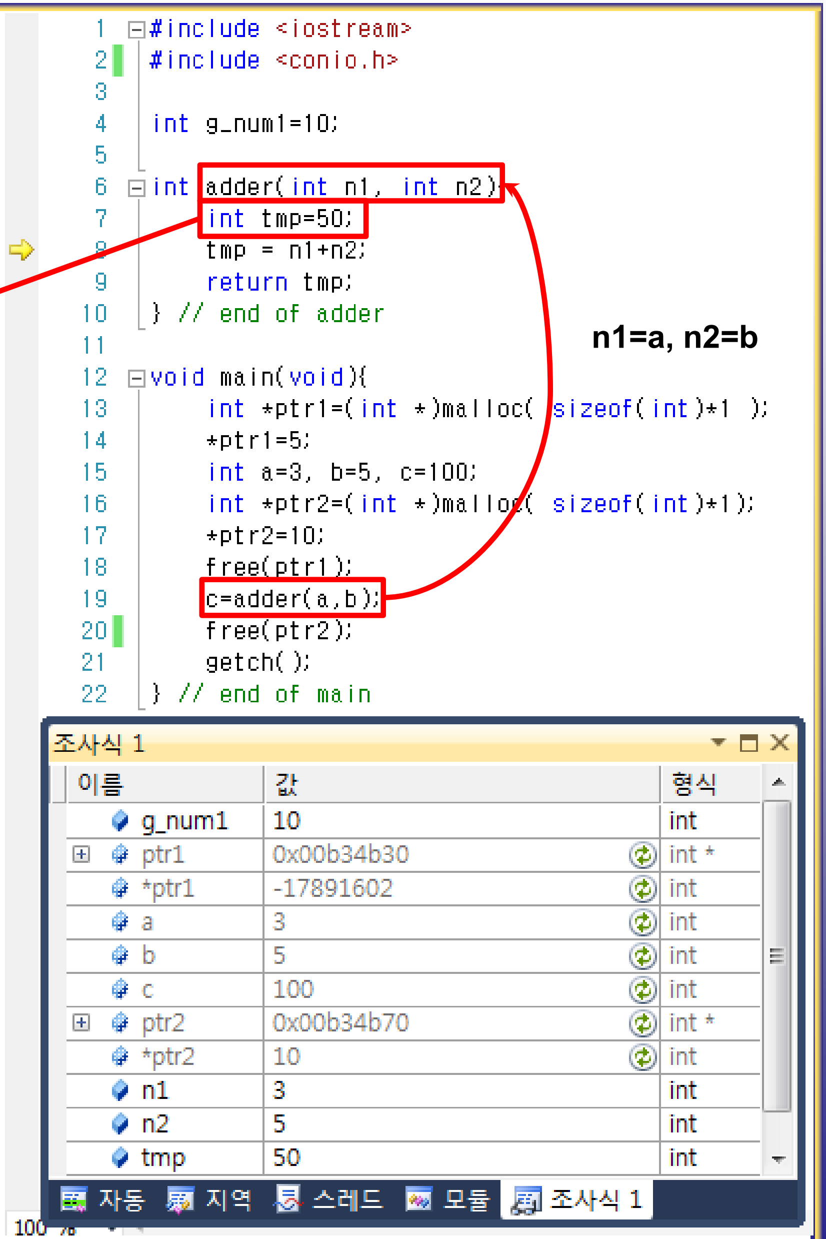
```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder(int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=(int *)malloc( sizeof(int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=(int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch();
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int
⊞ ptr1	0x00b34b30	int *
*ptr1	-17891602	int
a	3	int
b	5	int
c	100	int
⊞ ptr2	0x00b34b70	int *
*ptr2	10	int

Virtual memory(10/15)





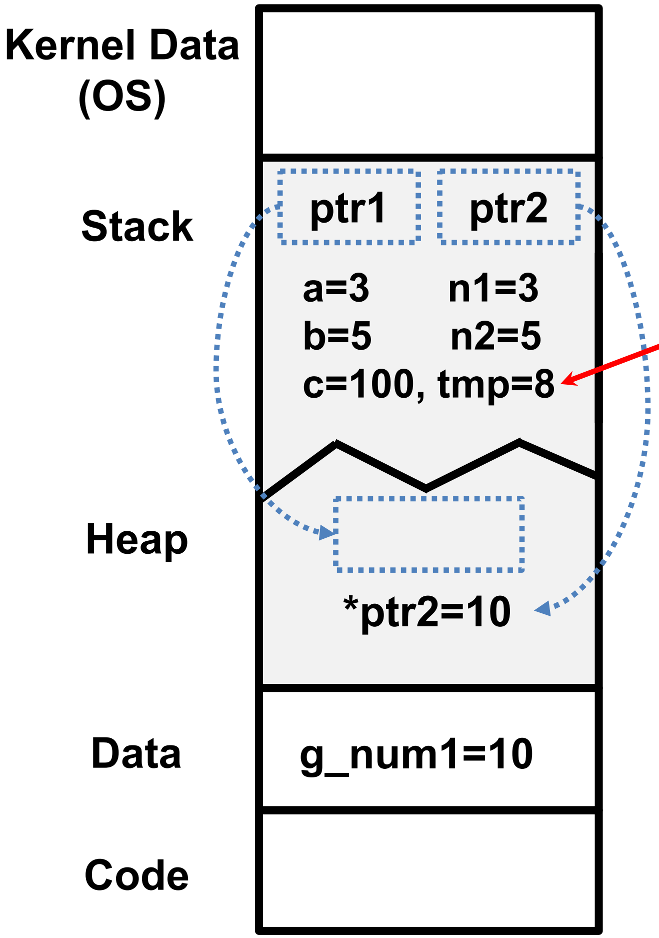
```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder( int n1, int n2)
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main( void){
    int *ptr1=( int *)malloc( sizeof( int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=( int *)malloc( sizeof( int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch( );
} // end of main
```

n1=a, n2=b

조사식 1

이름	값	형식
g_num1	10	int
⊞ ptr1	0x00b34b30	int *
*ptr1	-17891602	int
a	3	int
b	5	int
c	100	int
⊞ ptr2	0x00b34b70	int *
*ptr2	10	int
n1	3	int
n2	5	int
tmp	50	int

Virtual memory(11/15)



Kernel Data (OS)

Stack

ptr1 ptr2

a=3 n1=3

b=5 n2=5

c=100, tmp=8

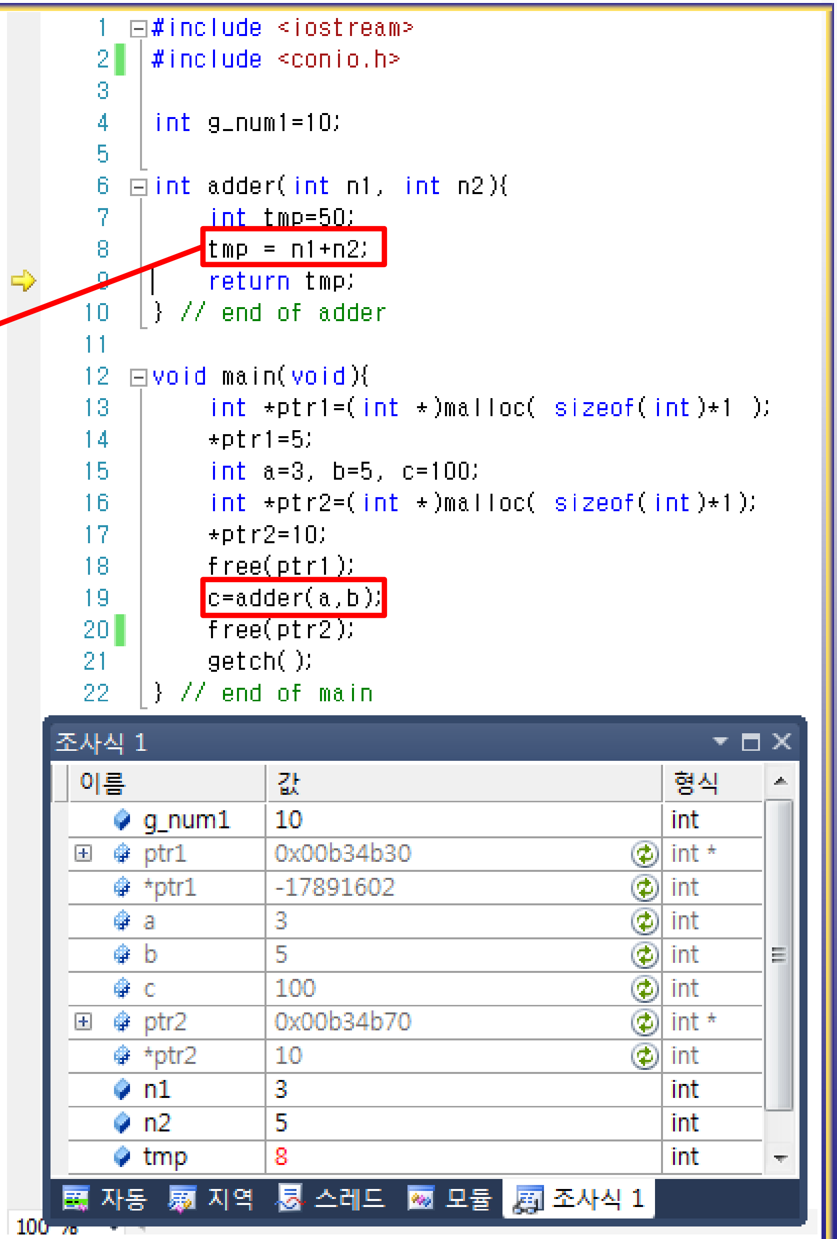
Heap

*ptr2=10

Data

g_num1=10

Code

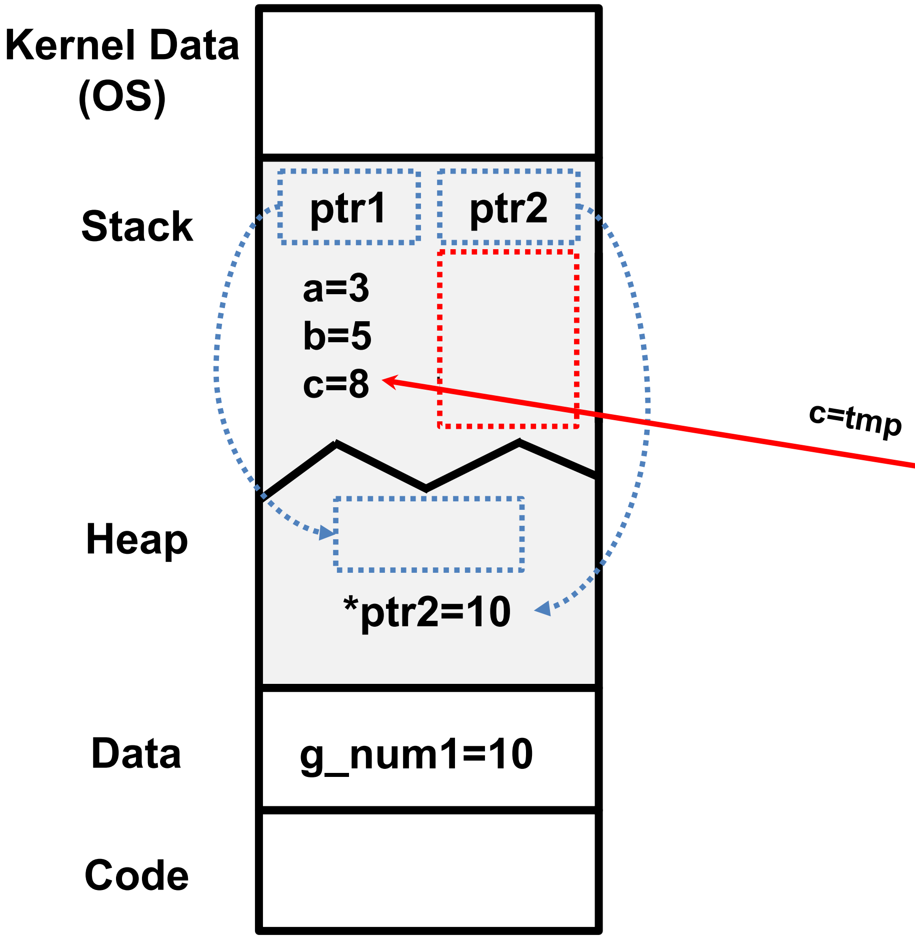


```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder( int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=(int *)malloc( sizeof(int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=(int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch( );
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int
ptr1	0x00b34b30	int *
*ptr1	-17891602	int
a	3	int
b	5	int
c	100	int
ptr2	0x00b34b70	int *
*ptr2	10	int
n1	3	int
n2	5	int
tmp	8	int

Virtual memory(12/15)



Kernel Data (OS)

Stack

ptr1

ptr2

a=3
b=5
c=8

Heap

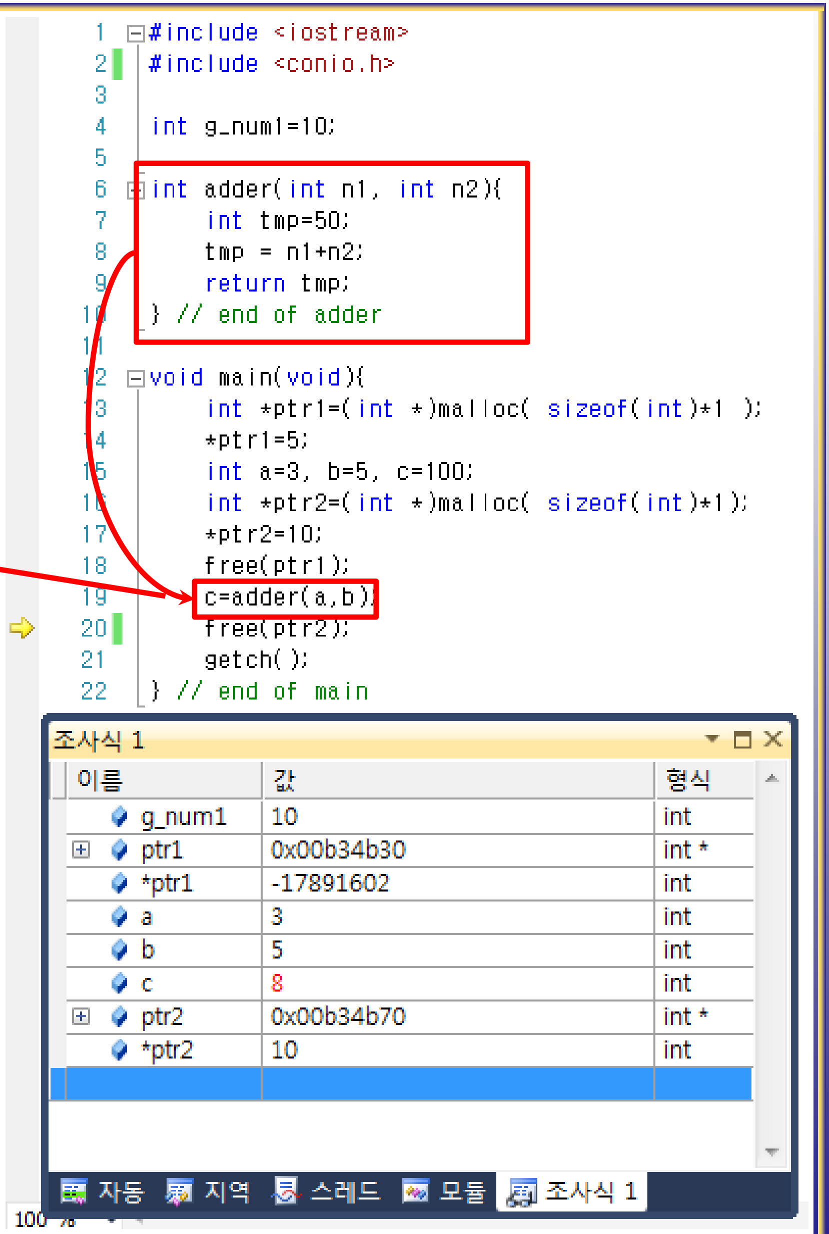
*ptr2=10

Data

g_num1=10

Code

c=tmp

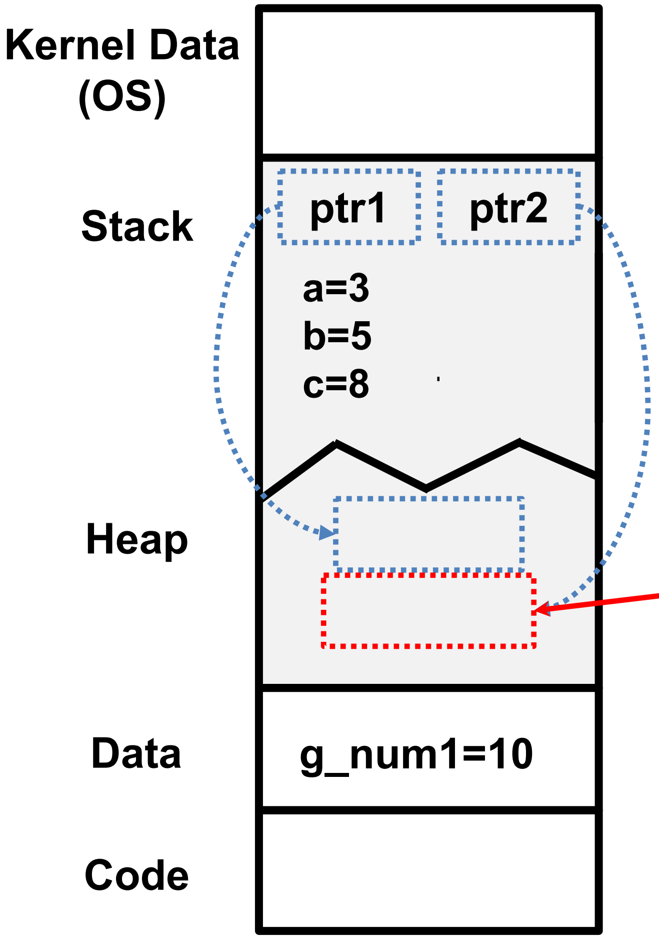


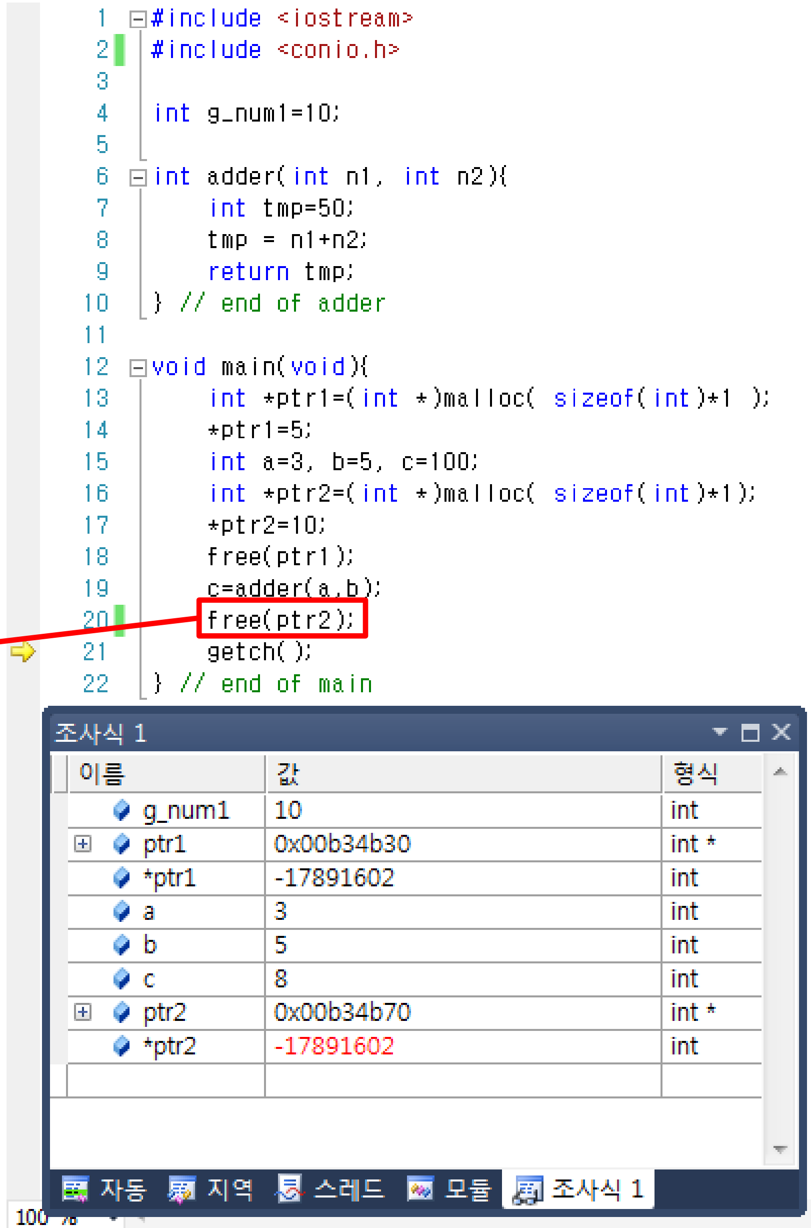
```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder(int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=(int *)malloc( sizeof(int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=(int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch( );
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int
⊞ ptr1	0x00b34b30	int *
*ptr1	-17891602	int
a	3	int
b	5	int
c	8	int
⊞ ptr2	0x00b34b70	int *
*ptr2	10	int

Virtual memory(13/15)



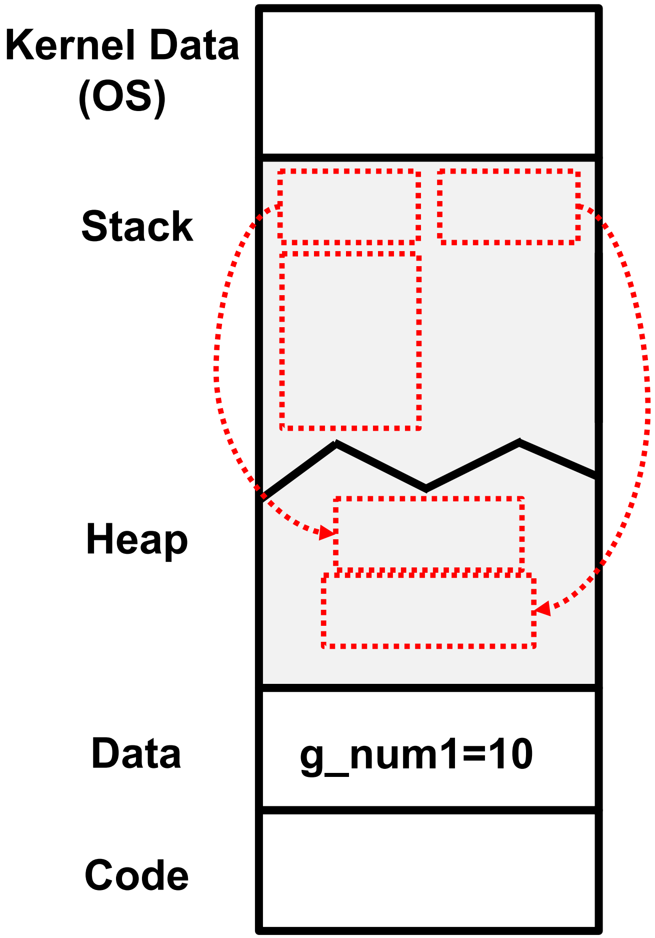


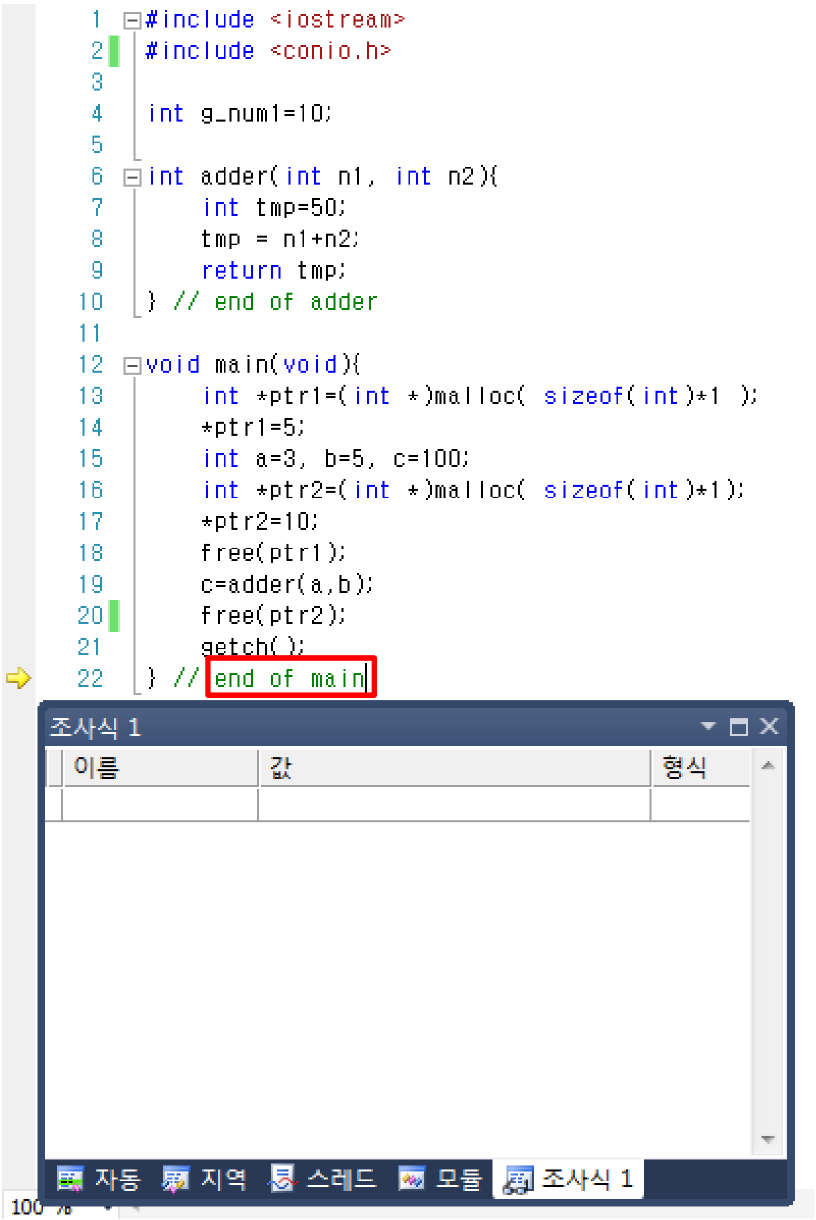
```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder( int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=(int *)malloc( sizeof(int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=(int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch( );
} // end of main
```

조사식 1

이름	값	형식
g_num1	10	int
⊞ ptr1	0x00b34b30	int *
*ptr1	-17891602	int
a	3	int
b	5	int
c	8	int
⊞ ptr2	0x00b34b70	int *
*ptr2	-17891602	int

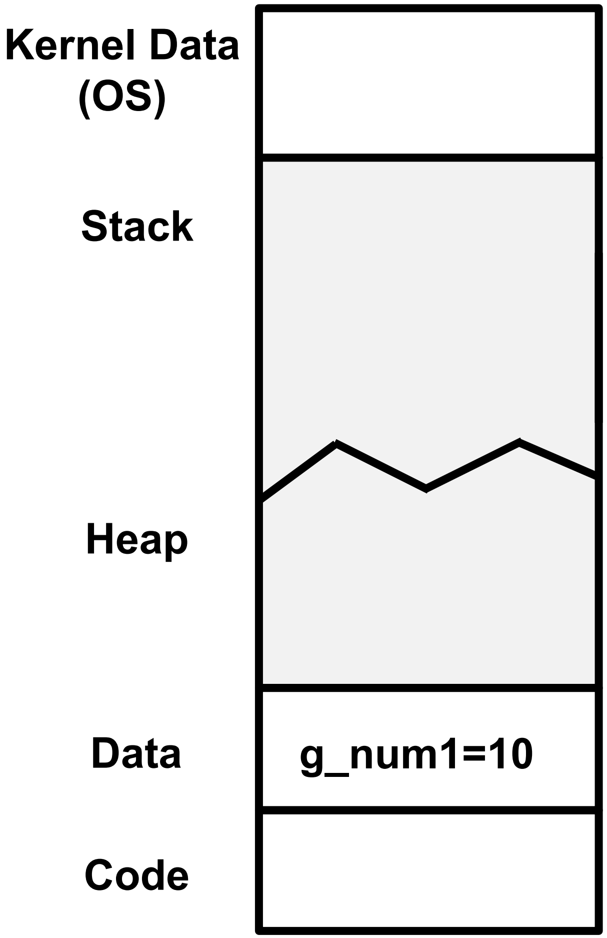
Virtual memory(14/15)

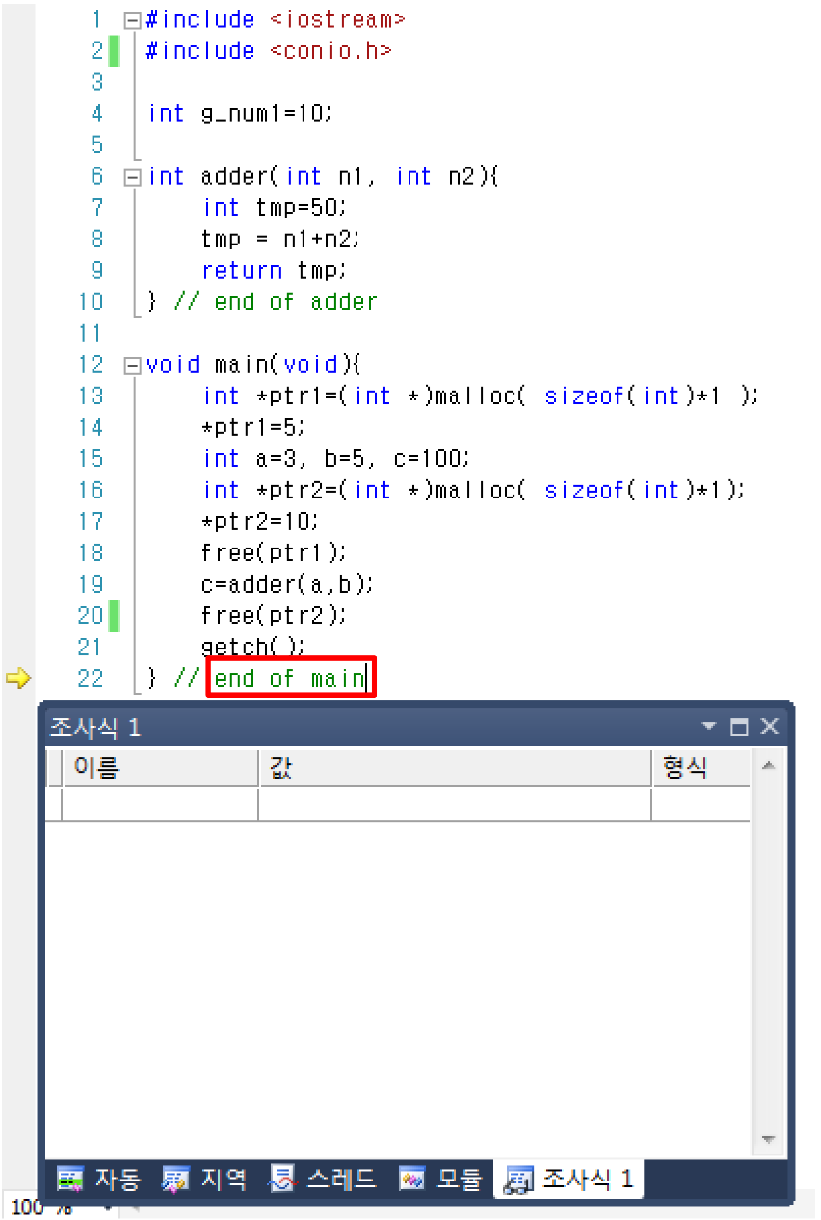




```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder(int n1, int n2){
     int tmp=50;
     tmp = n1+n2;
     return tmp;
} // end of adder

void main(void){
     int *ptr1=(int *)malloc( sizeof(int)*1 );
     *ptr1=5;
     int a=3, b=5, c=100;
     int *ptr2=(int *)malloc( sizeof(int)*1);
     *ptr2=10;
     free(ptr1);
     c=adder(a,b);
     free(ptr2);
     getch( );
} // end of main
```

Virtual memory(15/15)





```
#include <iostream>
#include <conio.h>

int g_num1=10;

int adder(int n1, int n2){
    int tmp=50;
    tmp = n1+n2;
    return tmp;
} // end of adder

void main(void){
    int *ptr1=(int *)malloc( sizeof(int)*1 );
    *ptr1=5;
    int a=3, b=5, c=100;
    int *ptr2=(int *)malloc( sizeof(int)*1);
    *ptr2=10;
    free(ptr1);
    c=adder(a,b);
    free(ptr2);
    getch( );
} // end of main
```