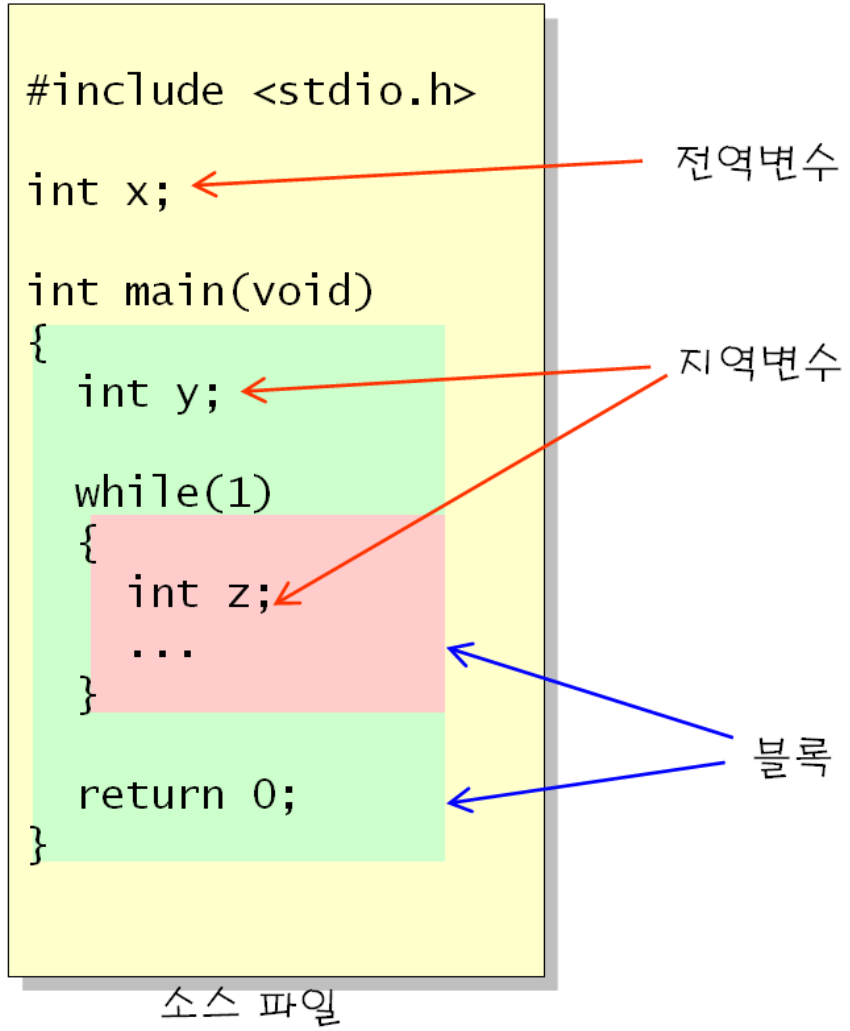
변수의 범위
scope of variables

2023

국민대학교 소프트웨어학부

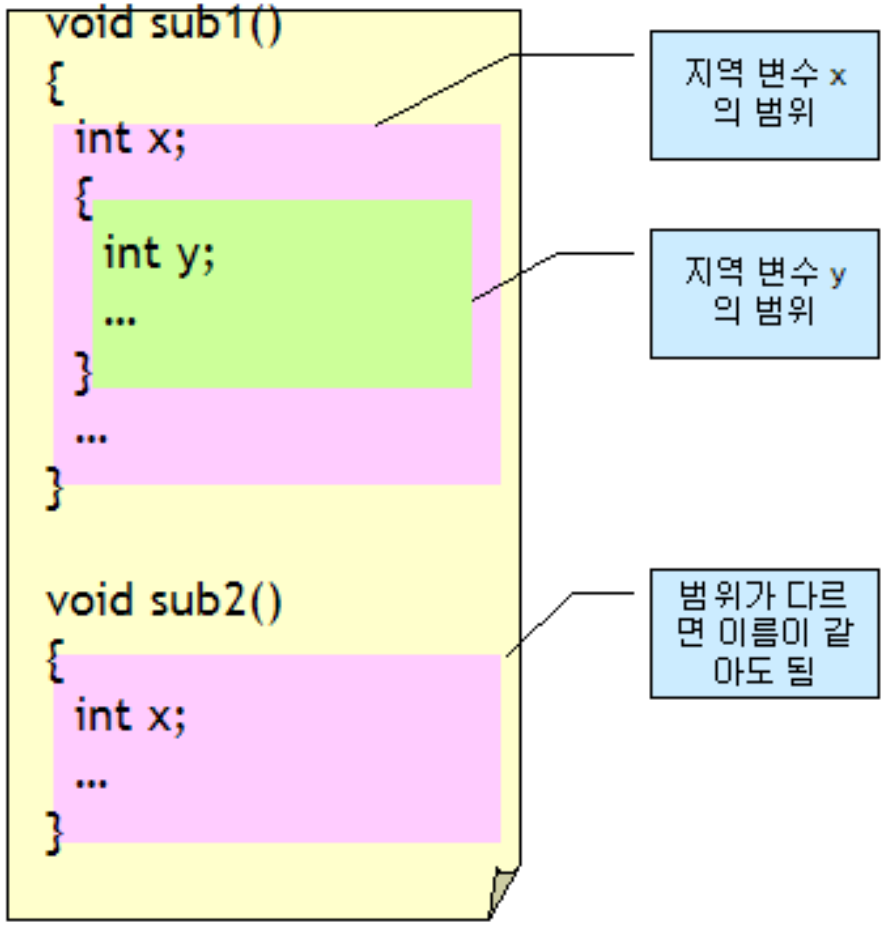
변수의 범위

- 범위(scope): 변수가 접근되고 사용되는 범위
- 변수의 범위는 변수가 선언되는 위치에 따라서 결정된다.
- 범위의 종류
  - 전역 변수: 함수의 외부에서 선언,
  - 지역 변수: 블록 안에서 선언

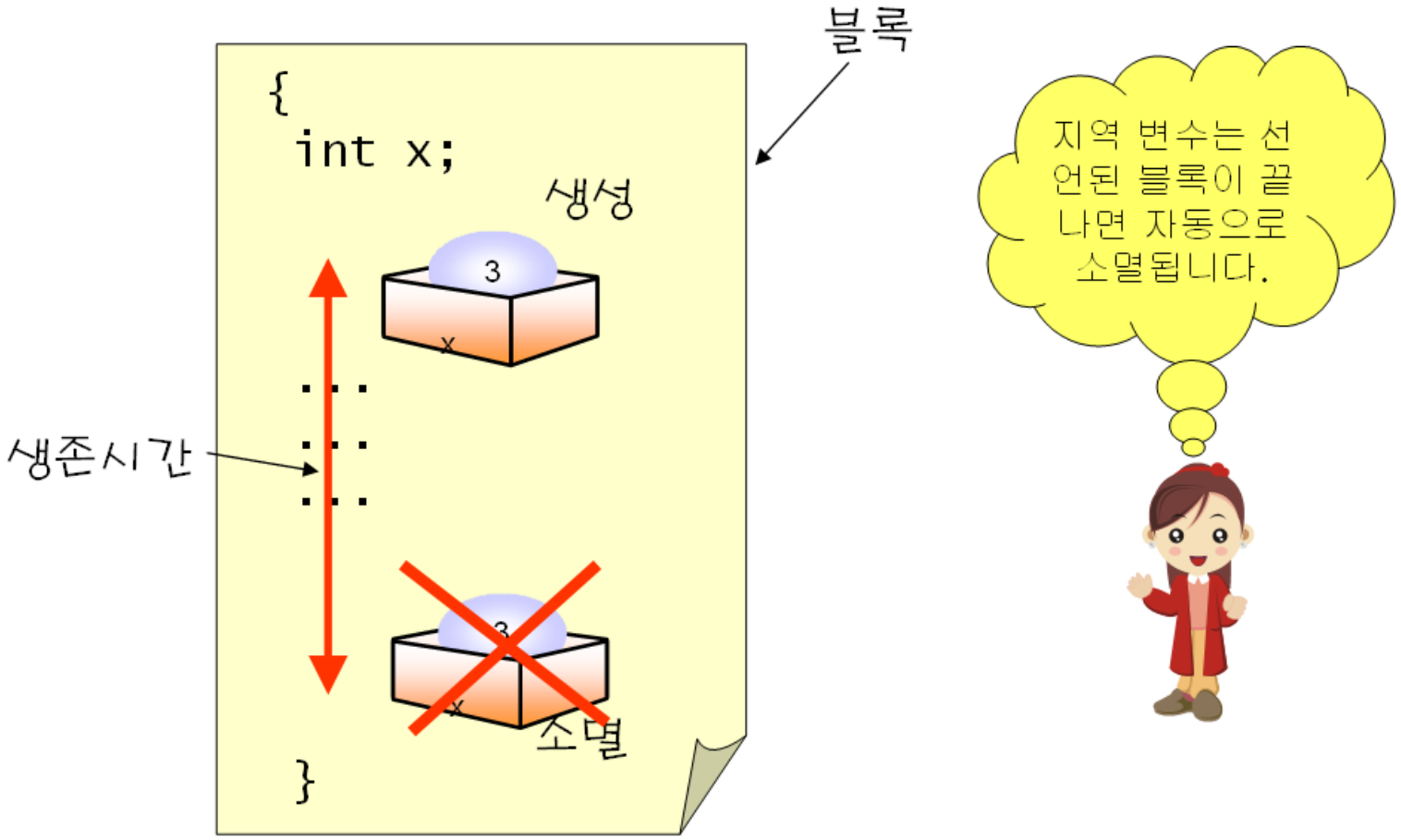


지역 변수의 scope

- 지역 변수(local variable): 블록 안에서 선언되는 변수



지역 변수의 생존 기간 lifetime



퀴즈 문제 중에서...

12.
```
for (int i = 0; i<4; i++){
  if (i%2)  continue;
  else      cout << i << endl;
}
```

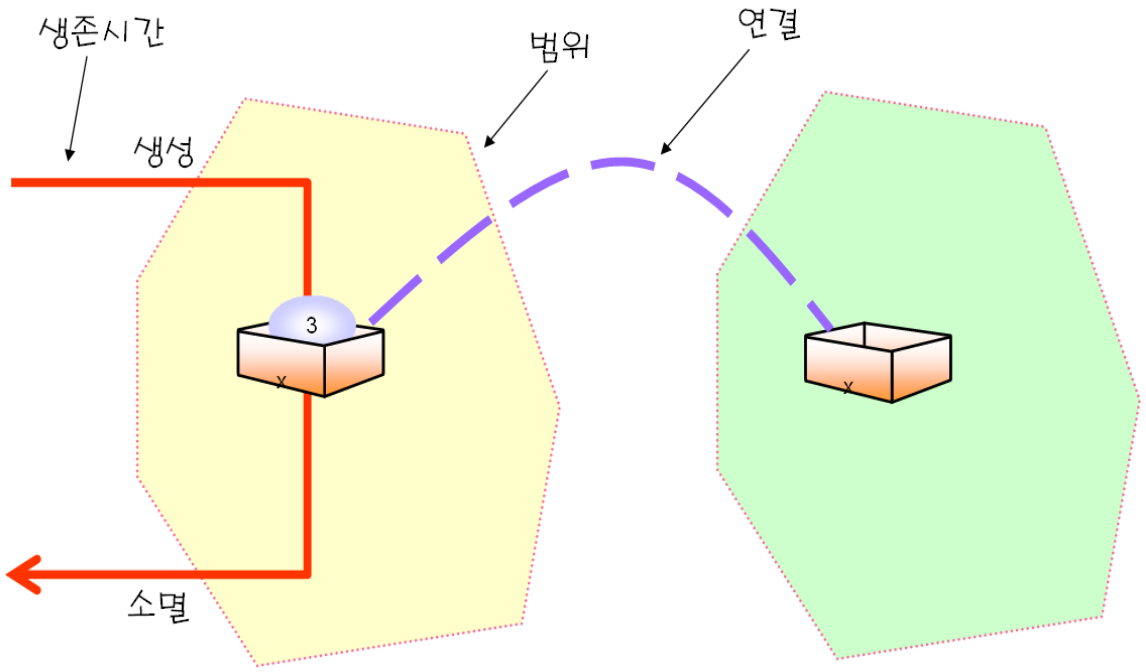
13.
```
for (int i = 0; i<4; i++){
  if (i%2)  continue;
  cout << i << endl;
}
```

14.
```
for (int i = 0; i<4; i++){
  if (i%2)  break;
  else      cout << i << endl;
}
```

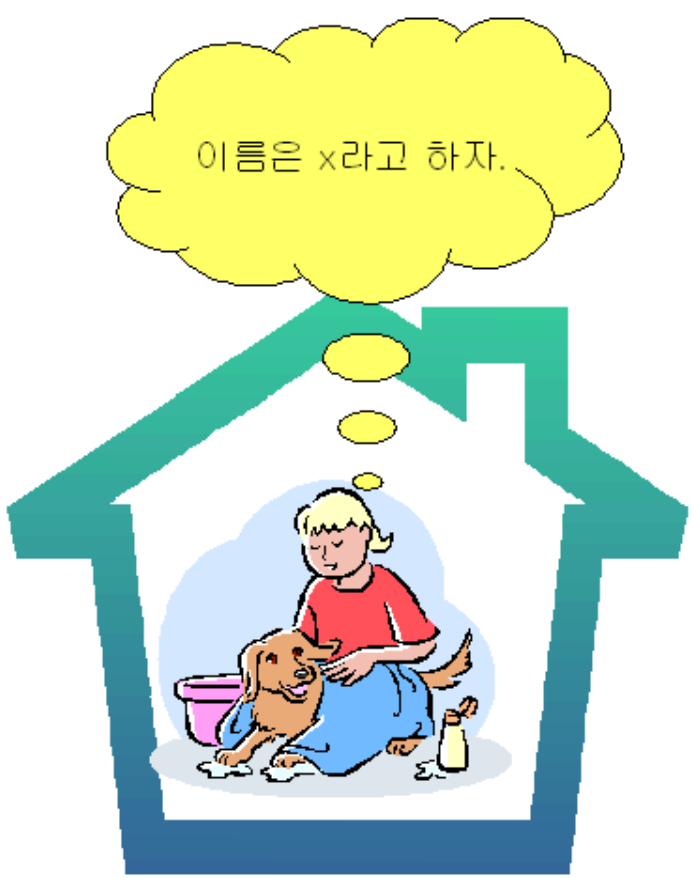
15.
```
for (int i = 0; i<4; i++){
  if (i%2)  continue;
  else      cout << i << endl;
}
cout << i << endl;
```

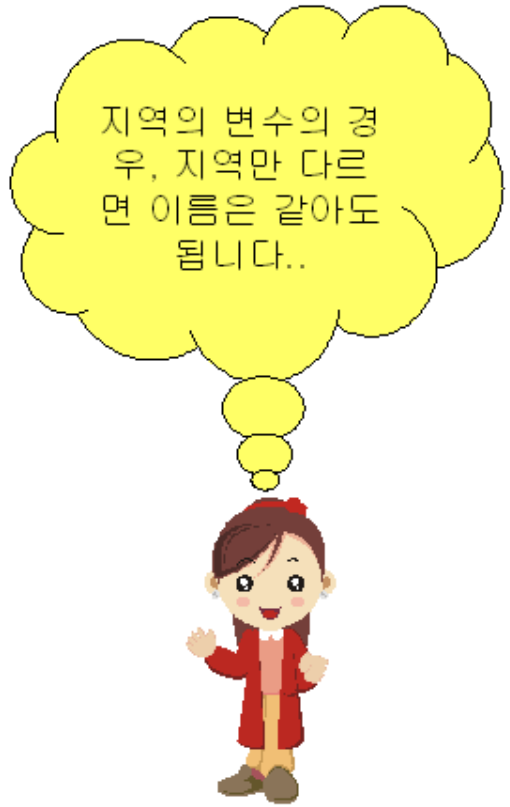
변수의 속성

- 변수의 속성 : 이름, 타입, 크기, 값 + 범위, 생존 시간, 연결
  - 범위(scope) : 변수가 사용 가능한 범위, 가시성
  - 생존 시간(lifetime): 메모리에 존재하는 시간
  - 연결(linkage): 다른 영역에 있는 변수와의 연결 상태

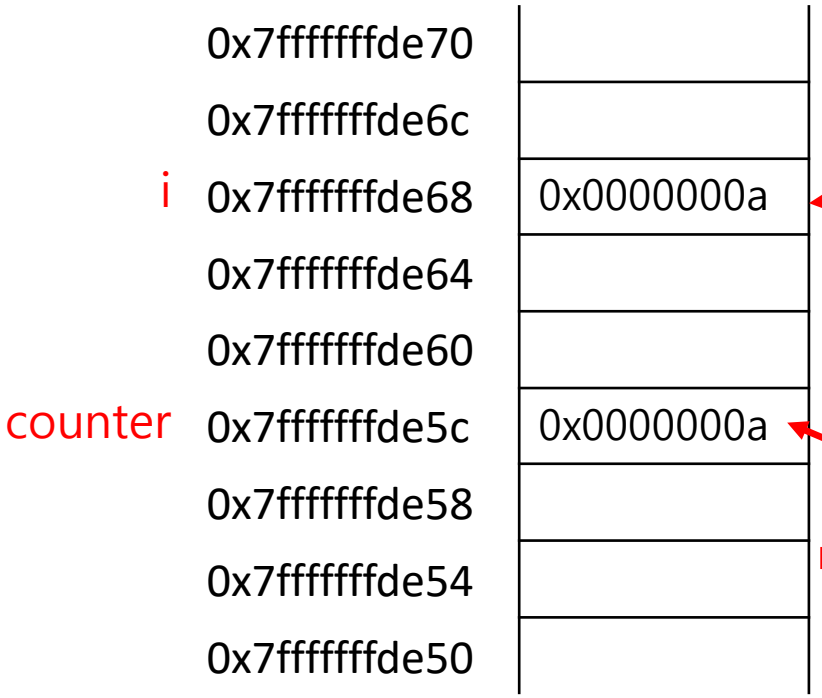


같은 이름의 지역 변수





함수의 매개 변수도 지역 변수




```
#include <iostream>
using namespace std;
int inc(int counter);
int main()
{
    int i;
    i = 10;
    cout << "함수 호출전 i=" << i << endl;
    inc(i);
    cout << "함수 호출후 i=" << i << endl;
    return 0;
}
int inc(int counter)
{
    counter++;
    return counter;
}
```

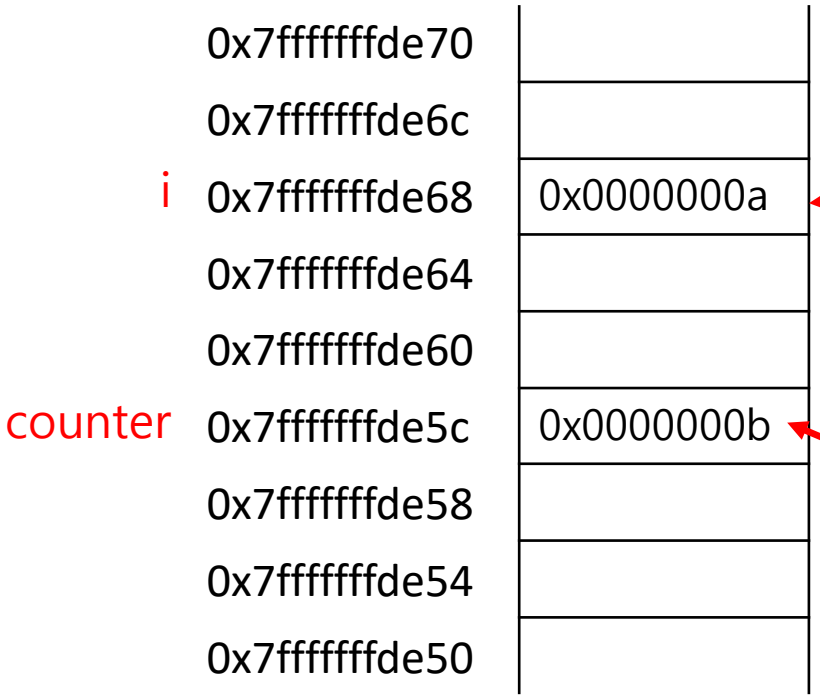
값에 의한 호출
(call by value)

매개 변수도 일종의 지역 변수임

int count = i;

```
함수 호출전 i=10
```

함수의 매개 변수



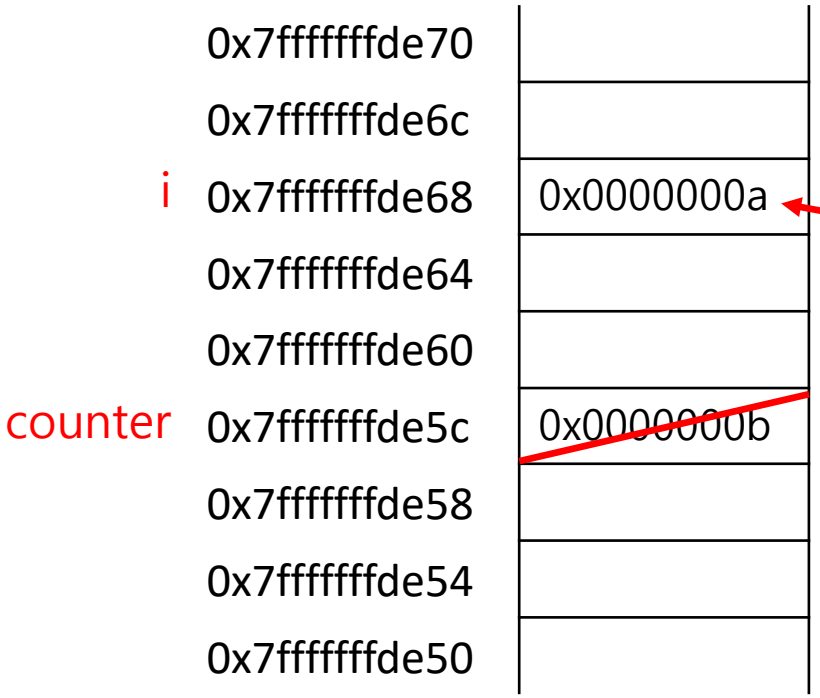

```
#include <iostream>
using namespace std;
int inc(int counter);
int main()
{
    int i;
    i = 10;
    cout << "함수 호출전 i=" << i << endl;
    inc(i);
    cout << "함수 호출후 i=" << i << endl;
    return 0;
}
int inc(int counter)
{
    counter++;
    return counter;
}
```

값에 의한 호출
(call by value)

매개 변수도 일종의 지역 변수임

```
함수 호출전 i=10
```

함수의 매개 변수




```
#include <iostream>
using namespace std;
int inc(int counter);
int main()
{
	int i;
	i = 10;
	cout << "함수 호출전 i=" << i << endl;
	inc(i);
	cout << "함수 호출후 i=" << i << endl;
	return 0;
}
int inc(int counter)
{
	counter++;
	return counter;
}
```

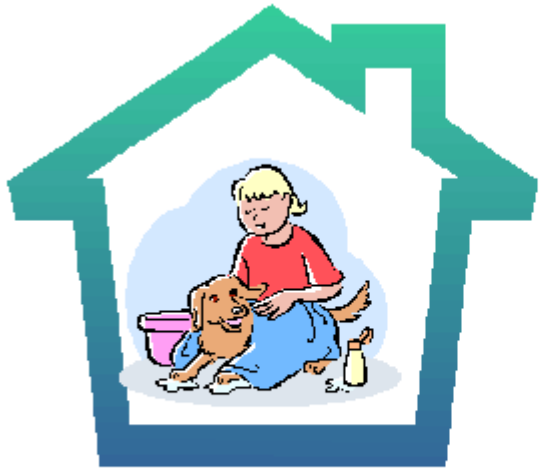
값에 의한 호출
(call by value)

매개 변수도 일종의 지역 변수임

```
함수 호출전 i=10
함수 호출후 i=10
```

전역 변수

- 전역 변수(global variable): 함수의 외부에서 선언되는 변수



지역변수

전역변수

전역 변수의 초기값과 생존 기간

```
// 전역 변수의 초기값과 생존 기간
#include <iostream>
using namespace std;
int counter;                    // 전역 변수
void set_counter(int i)
{
    counter = i;                // 직접 사용 가능
}
int main(void)
{
    cout << "counter=" << counter << endl;

    counter = 100;              // 직접 사용 가능
    cout << "counter=" << counter << endl;

    set_counter(20);
    cout << "counter=" << counter << endl;
    return 0;
}
```

```
counter=0
counter=100
counter=20
```

* 전역 변수의 초기값은 0

*생존 기간은 프로그램 시작부터 종료

전역 변수의 사용

```
// 전역 변수를 사용하여 프로그램이 복잡해지는 경우
#include <iostream>
using namespace std;
void f(void);
int i;
int main(void)
{
        for(i = 0;i < 5; i++)
        {
                f();
        }
        return 0;
}
void f(void)
{
        for(i = 0;i < 10; i++)
                cout << “#”;
}
```

##########

같은 이름의 전역 변수와 지역 변수

```
// 동일한 이름의 전역 변수와 지역 변수
#include <iostream>
using namespace std;

int sum = 1;                    // 전역 변수
int main()
{
    int i = 0;
    int sum = 0;                // 지역 변수

    for(i = 0; i <= 10; i++)
    {
        sum += i;               // 지역 변수가 전역 변수를 가린다.
    }
    cout << "sum = " << sum << endl;

    return 0;
}
```

지역 변수가 전역 변수를 가린다.

```
sum = 55
```

```
4  void f1(int a, int b=1);
5  int g1;
6  int garray[3];
7
8  int main(){
9    int   i=100, a[3];
10
11   {
12      int i;
13      i = 1;
14      a[0] = 20;
15   }
16   cout << "i = " << i << endl;
17
18   for (int i=0; i<3; i++) {}
19
20   cout << "i = " << i << endl;
21   f1(i);
22   return 0;
23 }
24 void f1(int a, int b){
25   int c = 100;
26   a = a+b+c;
27 }
```

```
Reading symbols from scope1...done.
(gdb) p g1
$1 = 0
(gdb) p garray
$2 = {0, 0, 0}
(gdb) p i
No symbol "i" in current context.
(gdb) b 14
Breakpoint 1 at 0x99f: file scope1.cpp, line 14.
(gdb) r
Starting program: /home/ejim/C2020/scope1

Breakpoint 1, main () at scope1.cpp:14
14              a[0] = 20;
(gdb) info locals
i = 1
i = 100
a = {21845, -8320, 32767}
(gdb) p i
$3 = 1
(gdb) p &i
$4 = (int *) 0x7fffffffde88
(gdb) n
16          cout << "i = " << i << endl;
(gdb) info locals
i = 100
a = {20, -8320, 32767}
(gdb) p &i
$5 = (int *) 0x7fffffffde84
```

프로그램이 수행되기 전에는 지역 변수는 볼 수 없다.

local 변수들을 모두 보여줌

```
4  void f1(int a, int b=1);
5  int g1;
6  int garray[3];
7
8  int main(){
9    int   i=100, a[3];
10
11   {
12      int i;
13      i = 1;
14      a[0] = 20;
15   }
16   cout << "i = " << i << endl;
17
18   for (int i=0; i<3; i++) {}
19
20   cout << "i = " << i << endl;
21   f1(i);
22   return 0;
23 }
24 void f1(int a, int b){
25   int c = 100;
26   a = a+b+c;
27 }
```

int a=i;

```
(gdb) n
i = 100
18          for (int i=0; i<3; i++) {}
(gdb)
20          cout << "i = " << i << endl;
(gdb)
i = 100
21          f1(i);
(gdb) s     step
f1 (a=100, b=1) at scope1.cpp:25
25          int c = 100;
(gdb) info locals
c = 32767
(gdb) bt
#0  f1 (a=100, b=1) at scope1.cpp:25
#1  0x000055555554a38 in main () at scope1.cpp:21
(gdb) p g1
$1 = 0
(gdb) p i
No symbol "i" in current context.
(gdb) p &a
$2 = (int *) 0x7fffffffde5c
(gdb) disp a
1: a = 100
(gdb) n
26          a = a+b+c;
1: a = 100
(gdb)
27        }
1: a = 201
```

0x7fffffffde84 (&i) 와 다름!

```
4  void f1(int a, int b=1);
5  int g1;
6  int garray[3];
7
8  int main(){
9    int   i=100, a[3];
10
11   {
12      int i;
13      i = 1;
14      a[0] = 20;
15   }
16   cout << "i = " << i << endl;
17
18   for (int i=0; i<3; i++) {}
19
20   cout << "i = " << i << endl;
21   f1(i);
22   return 0;
23 }
24 void f1(int a, int b){
25   int c = 100;
26   a = a+b+c;
27 }
```

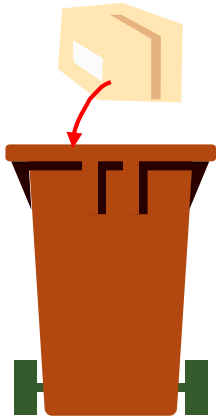
```
27       }
1: a = 201
(gdb) n
main () at scope1.cpp:22
22         return 0;
(gdb) p a
$3 = {20, -8320, 32767}
(gdb) p i
$4 = 100
(gdb) c
Continuing.
[Inferior 1 (process 6375) exited normally]
(gdb) p i
No symbol "i" in current context.
(gdb) p garray
$5 = {0, 0, 0}
(gdb) help s
Step program until it reaches a different source line.
Usage: step [N]
Argument N means step N times (or till program stops for another reason).
(gdb) help disp
Print value of expression EXP each time the program stops.
/FMT may be used before EXP as in the "print" command.
/FMT "i" or "s" or including a size-letter is allowed,
as in the "x" command, and then EXP is used to get the address to examine
and examining is done as in the "x" command.

With no argument, display all currently requested auto-display expressions.
Use "undisplay" to cancel display requests previously made.
```

프로그램 수행이 끝난 뒤 지역 변수는 볼 수 없다.

변수의 생존 기간

- 정적 할당(static allocation):
  - 프로그램 실행 시간 동안 계속 유지
  - 전역 변수는 정적 할당
- 자동 할당(automatic allocation):
  - 블록에 들어갈때 생성
  - 블록에서 나올때 소멸
  - 지역 변수는 자동 할당
- 생존 기간을 결정하는 요인
  - 변수가 선언된 위치
  - 저장 유형 지정자
- 저장 유형 지정자
  - static
  - extern

변수 생성



변수 소멸

저장 유형 지정자 static (for local variables)

```
#include <iostream>
using namespace std;
void sub(void);
int main()
{
	int i;
	for(i = 0;i < 5; i++)
		sub();
	return 0;
}
void sub(void)
{
	int auto_count = 0;
	static int static_count = 0;

	auto_count++;
	static_count++;
	cout << "auto_count=" << auto_count << endl;
	cout << "static_count=" << static_count << endl;
}
```

```
auto_count=1
static_count=1
auto_count=1
static_count=2
auto_count=1
static_count=3
...
```

정적 지역 변수로서
static을 붙이면 지역 변수가
정적 변수로 된다.

저장 유형 지정자 extern

extern1.c

컴파일러에게 변수가 다른 곳에서 선언되었음을 알린다

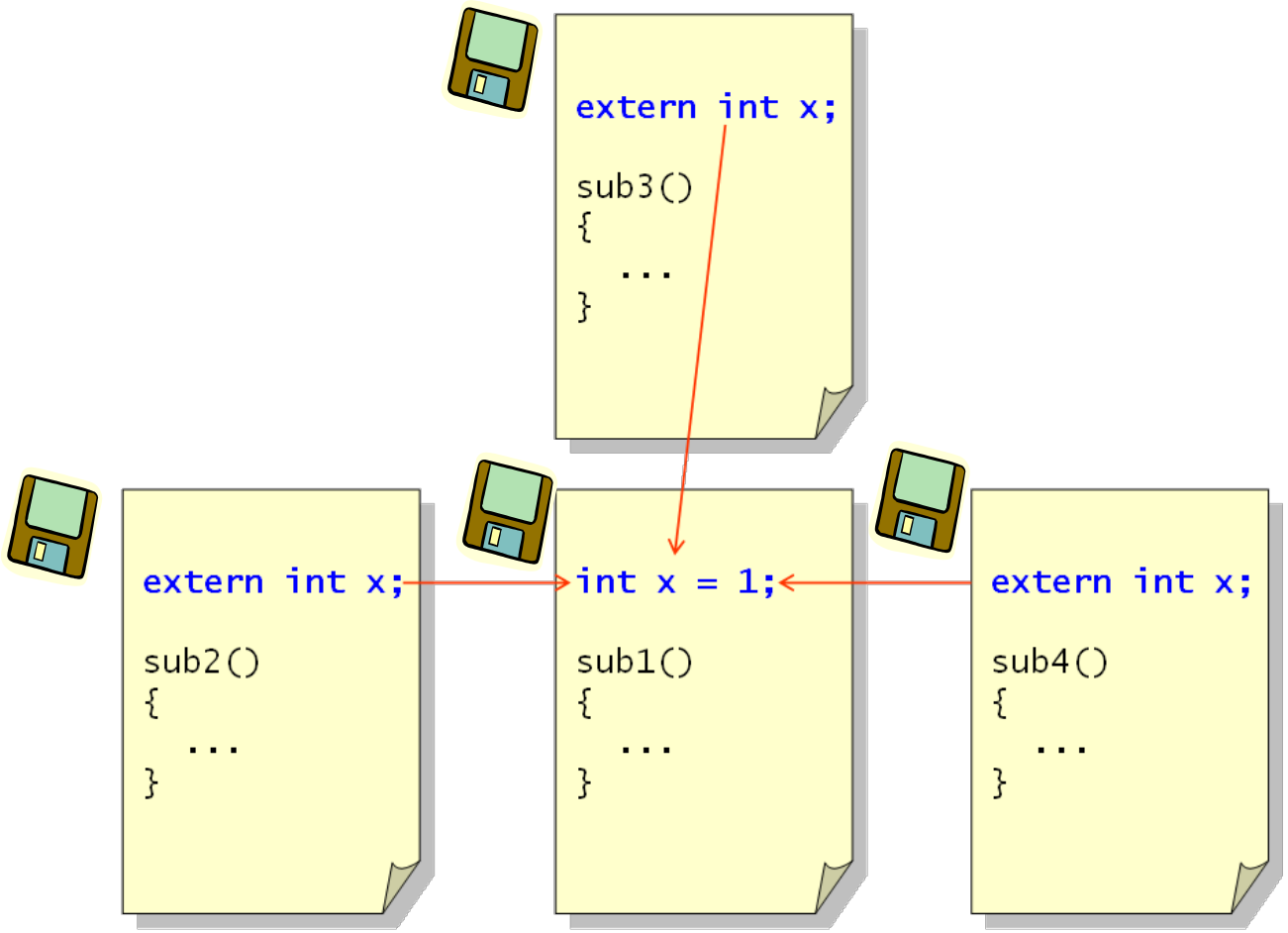
```
#include <iostream>
using namespace std;

int x;              // 전역 변수
extern int y;       // 현재 소스 파일의 뒷부분에 선언된 변수
extern int z;       // 다른 소스 파일의 변수
int main(void)
{
      extern int x;   // 전역 변수 x를 참조한다. 없어도된다.
      x = 10;
      y = 20;
      z = 30;

      return 0;
}
int y;       // 전역 변수
```

extern2.c

```
int z;
```

다중 소스 파일

- 연결은 흔히 다중 소스 파일에서 변수들을 연결하는데 사용된다.



```cpp
// scope2-1.cpp
#include <iostream>
using namespace std;

int f1(int a, int b=1);
int g1 = 3;
extern int garray[3]; //  = {4,5,6};

int main(){
  extern int   i;

  for (int j=0; j<2; j++){
     static int i = 1;
     i++;
     cout << "inside loop, i = " << i << endl;
  }
  cout << "outside loop, i = " << i << endl;

  cout << "f1(i)= " << f1(i)  << endl;
  cout << "f1(i)= " << f1(1000, 50)  << endl;
  return 0;
}
int f1(int a, int b){
  static int c = 100;
  c = a+b+c+g1;
  return c;
}
```

```cpp
// scope2-2.cpp
int garray[3];
int i;

void f2(void){
   extern int g1;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ g++ -g -o scope2 scope2-1.cpp scope2-2.cpp
ejim@ejim-VirtualBox:~/C2020$ ./scope2
inside loop, i = 2
inside loop, i = 3
outside loop, i = 0
f1(i)= 104
f1(i)= 1157
```

```cpp
// scope2-1.cpp
#include <iostream>
using namespace std;

int f1(int a, int b=1);
int g1 = 3;
extern int garray[3]; //  = {4,5,6};

int main(){
  extern int   i;

  for (int j=0; j<2; j++){
     static int i = 1;
     i++;
     cout << "inside loop, i = " << i << e
  }
  cout << "outside loop, i = " << i << end

  cout << "f1(i)= " << f1(i)  << endl;
  cout << "f1(i)= " << f1(1000, 50)  << en
  f2();
  return 0;
}

int f1(int a, int b){
  static int c = 100;
  c = a+b+c+g1;
  return c;
}
```

```cpp
// scope2-2.cpp
#include <iostream>
using namespace std;

int garray[3];
int i;

void f2(void){
   extern int g1;
   cout << "f1(10,20) = " << f1(10, 20) << endl;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ g++ -g -o scope2 scope2-1.cpp scope2-2.cpp
scope2-1.cpp: In function ‘int main()’:
scope2-1.cpp:21:3: error: ‘f2’ was not declared in this scope
   f2();
   ^~
scope2-1.cpp:21:3: note: suggested alternative: ‘f1’
   f2();
   ^~
   f1
scope2-2.cpp: In function ‘void f2()’:
scope2-2.cpp:10:30: error: ‘f1’ was not declared in this scope
    cout << "f1(10,20) = " << f1(10, 20) << endl;
                             ^~
scope2-2.cpp:10:30: note: suggested alternative: ‘g1’
    cout << "f1(10,20) = " << f1(10, 20) << endl;
                             ^~
                             g1
```

```cpp
// scope2-1.cpp
#include <iostream>
using namespace std;
void f2(void);
int f1(int a, int b=1);
int g1 = 3;
extern int garray[3]; //  = {4,5,6};

int main(){
  extern int   i;

  for (int j=0; j<2; j++){
     static int i = 1;
     i++;
     cout << "inside loop, i = " << i << endl;
  }
  cout << "outside loop, i = " << i << endl;

  cout << "f1(i)= " << f1(i)  << endl;
  cout << "f1(i)= " << f1(1000, 50)  << endl;
  f2();
  return 0;
}
int f1(int a, int b){
  static int c = 100;
  c = a+b+c+g1;
  return c;
}
```

```cpp
// scope2-2.cpp
#include <iostream>
using namespace std;
int f1(int a, int b=1);
int garray[3];
int i;

void f2(void){
   extern int g1;
   cout << "f1(g1,20) = " << f1(g1, 20) << endl;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./scope2
inside loop, i = 2
inside loop, i = 3
outside loop, i = 0
f1(i)= 104
f1(i)= 1157
f1(g1,20) = 1183
```

```cpp
// scope2-1.cpp
#include <iostream>
using namespace std;
void f2(void);
static int f1(int a, int b=1);
static int g1 = 3;
extern int garray[3]; //  = {4,5,6};

int main(){
  extern int   i;

  for (int j=0; j<2; j++){
     static int i = 1;
     i++;
     cout << "inside loop, i = " << i << endl;
  }
  cout << "outside loop, i = " << i << endl;

  cout << "f1(i)= " << f1(i)  << endl;
  cout << "f1(i)= " << f1(1000, 50)  << endl;
  f2();
  return 0;
}
int f1(int a, int b){
  static int c = 100;
  c = a+b+c+g1;
  return c;
}
```

전역 변수와 함수를 static 으로 선언하면?

```cpp
// scope2-2.cpp
#include <iostream>
using namespace std;
int f1(int a, int b=1);
int garray[3];
int i;

void f2(void){
   extern int g1;
   cout << "f1(g1,20) = " << f1(g1, 20) << endl;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ g++ -g -o scope2 scope2-1.cpp scope2-2.cpp
/tmp/ccsY0pC3.o: In function `f2()':
/home/ejim/C2020/scope2-2.cpp:10: undefined reference to `g1'
/home/ejim/C2020/scope2-2.cpp:10: undefined reference to `f1(int, int)'
collect2: error: ld returned 1 exit status
```

저장 유형 지정자 static (for global variables & functions)

- 전역 변수나 함수의 static 선언은 지역 변수 static 선언과 의미가 다르다.
- 선언된 file 외부에서 접근할 수 없다는 뜻이다.

static global variable

linkage1.c

```
#include <iostream>
using namespace std;
int all_files;              // 다른 소스 파일에서도 사용할 수 있는 전역 변수
static int this_file; // 현재의 소스 파일에서만 사용할 수 있는 전역 변수
extern void sub();

int main()
{
        sub();
        cout << all_files << endl;
        return 0;
}
```

연결

linkage2.c

```
extern int all_files;
void sub()
{
        all_files = 10;
}
```

```
10
```

static function

main.c

```
#include <iostream>
using namespace std;
extern void f2();
int main(void)
{
	f2();
	return 0;
}
```

static 함수는 파일 안에서만 사용할 수 있다

sub.c

```
#include <iostream>
using namespace std;
static void f1()
{
	cout << "f1()이 호출되었습니다.\n";
}

void f2()
{
	f1();
	cout << "f2()가 호출되었습니다.\n";
}
```

```cpp
// scope2-1.cpp
#include <iostream>
using namespace std;

int f1(int a, int b=1);
int g1 = 3;
extern int garray[3]; //  = {4,5,6};

int main(){
  extern int   i;

  for (int j=0; j<2; j++){
     static int i = 1;
     i++;
     cout << "inside loop, i = " << i << e
  }
  cout << "outside loop, i = " << i << end

  cout << "f1(i)= " << f1(i)  << endl;
  cout << "f1(i)= " << f1(1000, 50)  << en
  f2();
  return 0;
}

int f1(int a, int b){
  static int c = 100;
  c = a+b+c+g1;
  return c;
}
```

```cpp
// scope2-2.cpp
#include <iostream>
using namespace std;

int garray[3];
int i;

void f2(void){
   extern int g1;
   cout << "f1(10,20) = " << f1(10, 20) << endl;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ g++ -g -o scope2 scope2-1.cpp scope2-2.cpp
scope2-1.cpp: In function ‘int main()’:
scope2-1.cpp:21:3: error: ‘f2’ was not declared in this scope
   f2();
   ^~
scope2-1.cpp:21:3: note: suggested alternative: ‘f1’
   f2();
   ^~
   f1
scope2-2.cpp: In function ‘void f2()’:
scope2-2.cpp:10:30: error: ‘f1’ was not declared in this scope
    cout << "f1(10,20) = " << f1(10, 20) << endl;
                             ^~
scope2-2.cpp:10:30: note: suggested alternative: ‘g1’
    cout << "f1(10,20) = " << f1(10, 20) << endl;
                             ^~
                             g1
```

```cpp
// scope2-1.cpp
#include <iostream>
using namespace std;
void f2(void);
int f1(int a, int b=1);
int g1 = 3;
extern int garray[3]; //  = {4,5,6};

int main(){
  extern int   i;

  for (int j=0; j<2; j++){
     static int i = 1;
     i++;
     cout << "inside loop, i = " << i << endl;
  }
  cout << "outside loop, i = " << i << endl;

  cout << "f1(i)= " << f1(i)  << endl;
  cout << "f1(i)= " << f1(1000, 50)  << endl;
  f2();
  return 0;
}
int f1(int a, int b){
  static int c = 100;
  c = a+b+c+g1;
  return c;
}
```

```cpp
// scope2-2.cpp
#include <iostream>
using namespace std;
int f1(int a, int b=1);
int garray[3];
int i;

void f2(void){
   extern int g1;
   cout << "f1(g1,20) = " << f1(g1, 20) << endl;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./scope2
inside loop, i = 2
inside loop, i = 3
outside loop, i = 0
f1(i)= 104
f1(i)= 1157
f1(g1,20) = 1183
```

function prototype 은 헤더파일로 만드는 것이 좋다.

```cpp
// scope2-1.cpp
#include <iostream>
#include "scope2-1.h"
#include "scope2-2.h"
using namespace std;

int g1 = 3;
extern int garray[3]; //  = {4,5,6};

int main(){
  extern int   i;

  for (int j=0; j<2; j++){
     static int i = 1;
     i++;
     cout << "inside loop, i = " << i << endl;
  }
  cout << "outside loop, i = " << i << endl;

  cout << "f1(i)= " << f1(i)  << endl;
  cout << "f1(i)= " << f1(1000, 50) << endl;
  f2();
  return 0;
}
int f1(int a, int b){
  static int c = 100;
  c = a+b+c+g1;
  return c;
}
```

```cpp
// scope2-1.h
int f1(int a, int b=1);
```

```cpp
// scope2-2.h
void f2(void);
```

```cpp
// scope2-2.cpp
#include <iostream>
#include "scope2-1.h"
#include "scope2-2.h"
using namespace std;

int garray[3];
int i;

void f2(void){
   extern  int g1;
   cout << "f1(g1,20) = " << f1(g1, 20) << endl;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ g++ -g -o scope2 scope2-1.cpp scope2-2.cpp
ejim@ejim-VirtualBox:~/C2020$ ./scope2
inside loop, i = 2
inside loop, i = 3
outside loop, i = 0
f1(i)= 104
f1(i)= 1157
f1(g1,20) = 1183
```