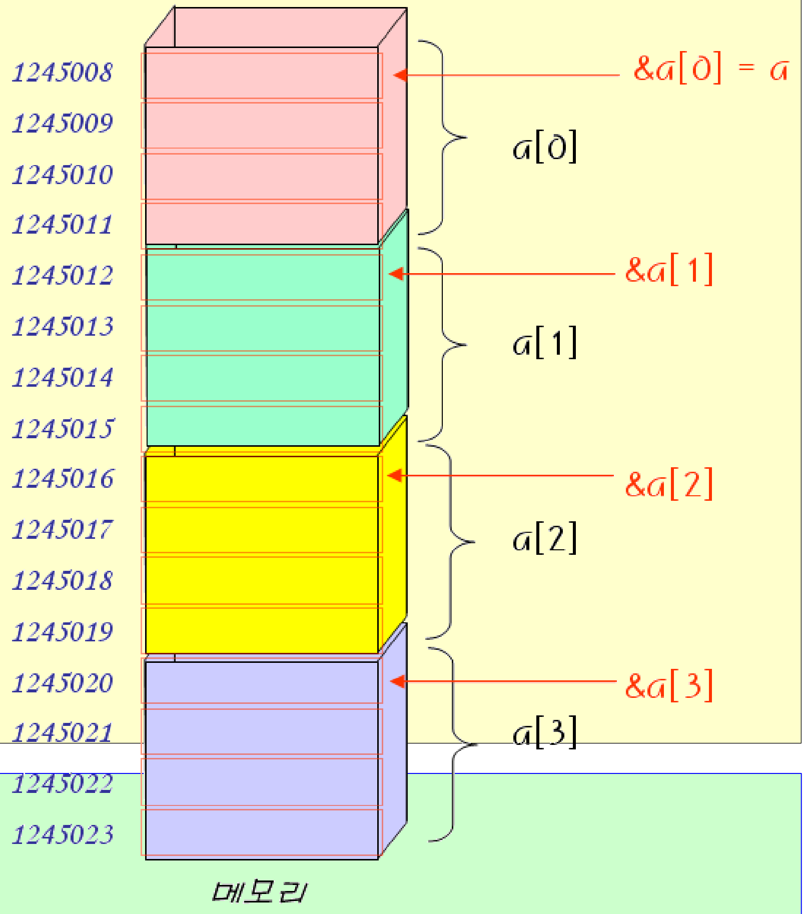
메모리의 동적 할당
dynamic allocation

2023

국민대학교 소프트웨어학부

포인터와 배열 : 배열의 이름은 포인터로 취급

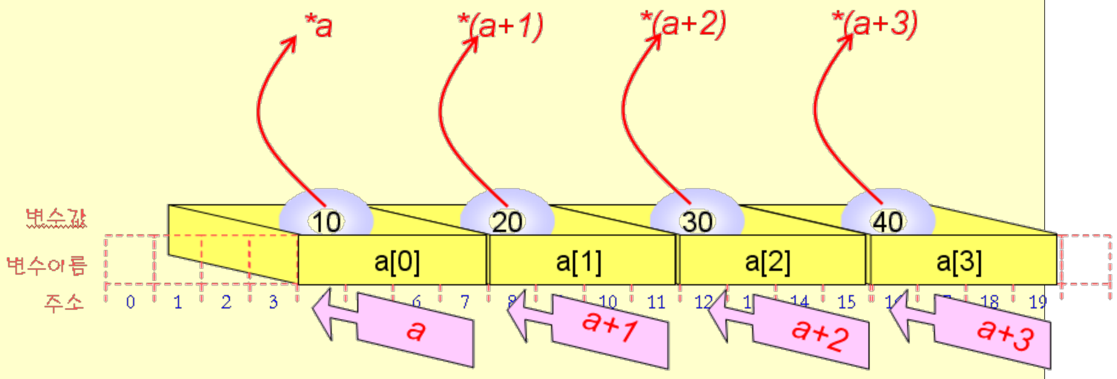
```
#include <iostream>
using namespace std;
int main()
{
        int a[] = { 10, 20, 30, 40, 50 };

        cout << "&a[0] = " << &a[0] << endl;
        cout << "&a[1] = " << &a[1] << endl;
        cout << "&a[2] = " << &a[2] << endl;

        cout << "a = " << a << endl;

        return 0;
}
```



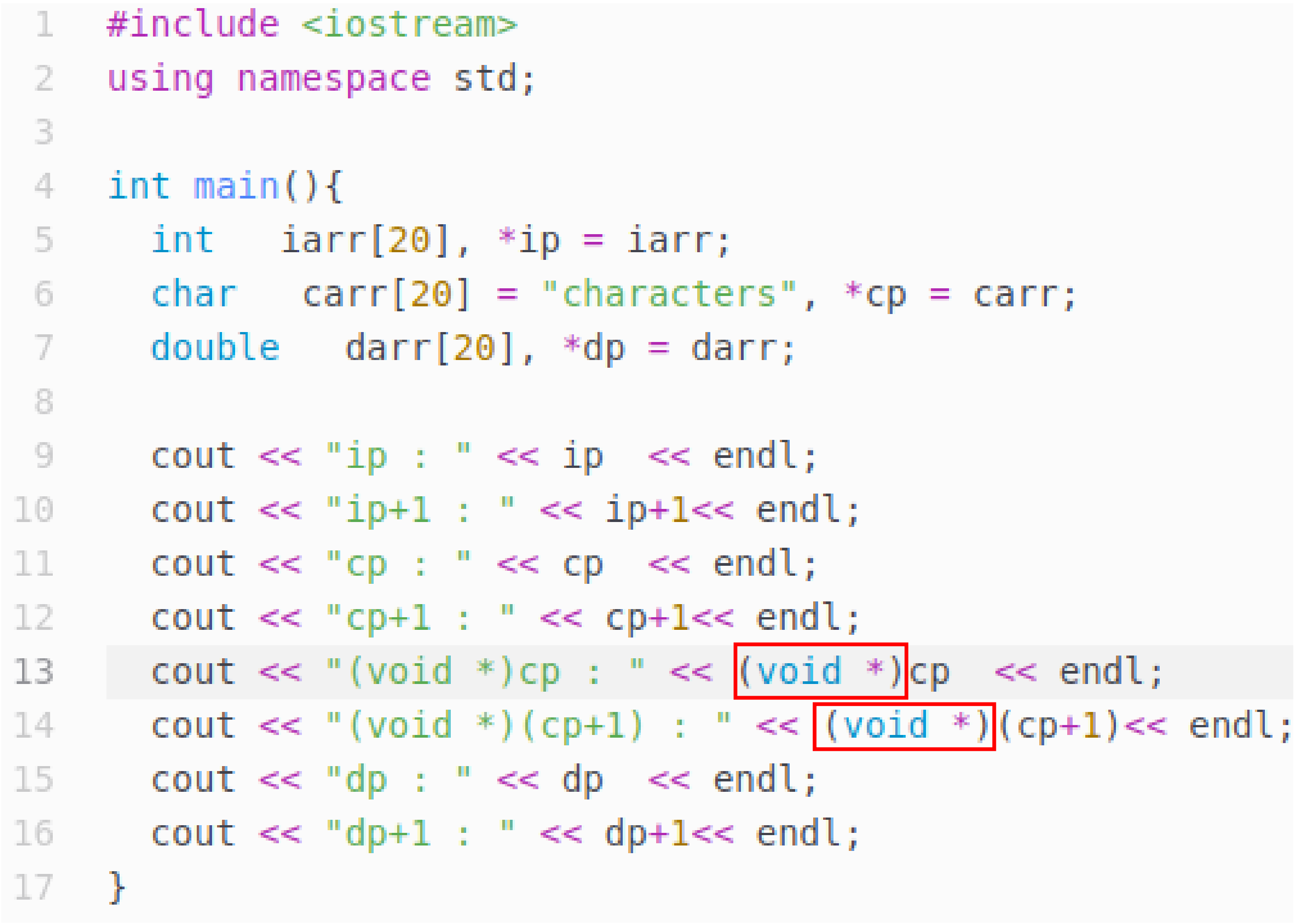
```
&a[0] = 1245008
&a[1] = 1245012
&a[2] = 1245016
a = 1245008
```

포인터 연산

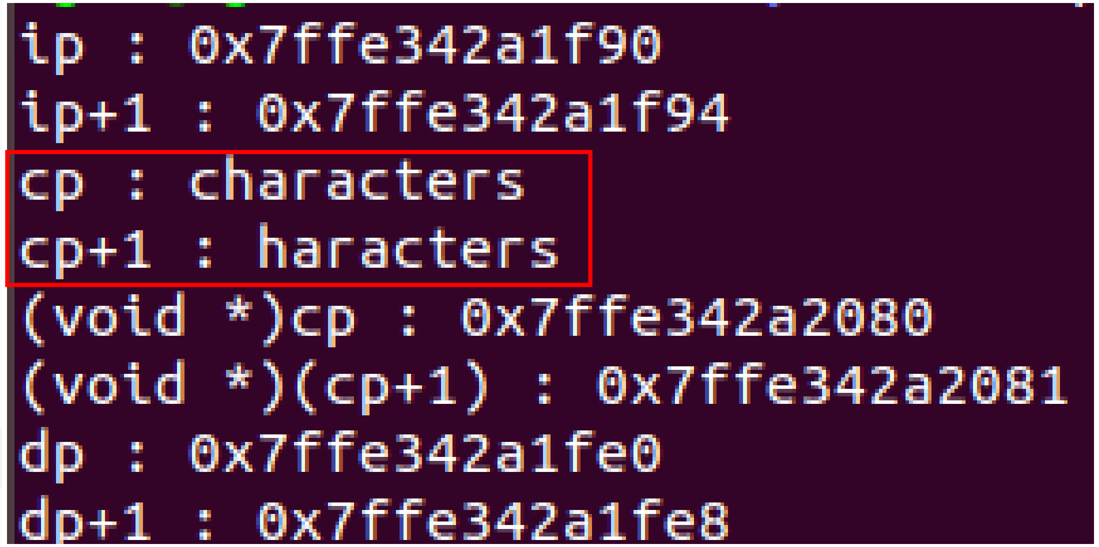
```
#include <iostream>
using namespace std;
int main()
{
	int a[] = { 10, 20, 30, 40, 50 };

	cout << "a = " << a << endl;
	cout << "a + 1 = " << a+1 << endl;
	cout << "*a = " << *a << endl;
	cout << "*(a+1) " << *(a+1) << endl;
	return 0;
}
```



```
a = 1245008
a + 1 = 1245012
*a = 10
*(a+1) = 20
```



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

int main(){
  int   iarr[20], *ip = iarr;
  char   carr[20] = "characters", *cp = carr;
  double   darr[20], *dp = darr;

  cout << "ip : " << ip  << endl;
  cout << "ip+1 : " << ip+1<< endl;
  cout << "cp : " << cp  << endl;
  cout << "cp+1 : " << cp+1<< endl;
  cout << "(void *)cp : " << (void *)cp  << endl;
  cout << "(void *)(cp+1) : " << (void *)(cp+1)<< endl;
  cout << "dp : " << dp  << endl;
  cout << "dp+1 : " << dp+1<< endl;
}
```



```
ip : 0x7ffe342a1f90
ip+1 : 0x7ffe342a1f94
cp : characters
cp+1 : haracters
(void *)cp : 0x7ffe342a2080
(void *)(cp+1) : 0x7ffe342a2081
dp : 0x7ffe342a1fe0
dp+1 : 0x7ffe342a1fe8
```

포인터를 사용한 방법의 장점

- 인덱스 표기법보다 빠르다.
  - 원소의 주소를 계산할 필요가 없다.

```
int get_sum1(int a[], int n)
{
	int i;
	int sum = 0;

	for(i = 0; i < n; i++ )
		sum += a[i];
	return sum;
}
```

인덱스 표기법 사용

```
int get_sum2(int a[], int n)
{
	int i;
	int *p;
	int sum = 0;

	p = a;
	for(i = 0; i < n; i++ )
		sum += *p++;
	return sum;
}
```

포인터 사용

동적 할당 메모리의 개념

- 프로그램이 메모리를 할당받는 방법
  - 정적(static)
  - 동적(dynamic)

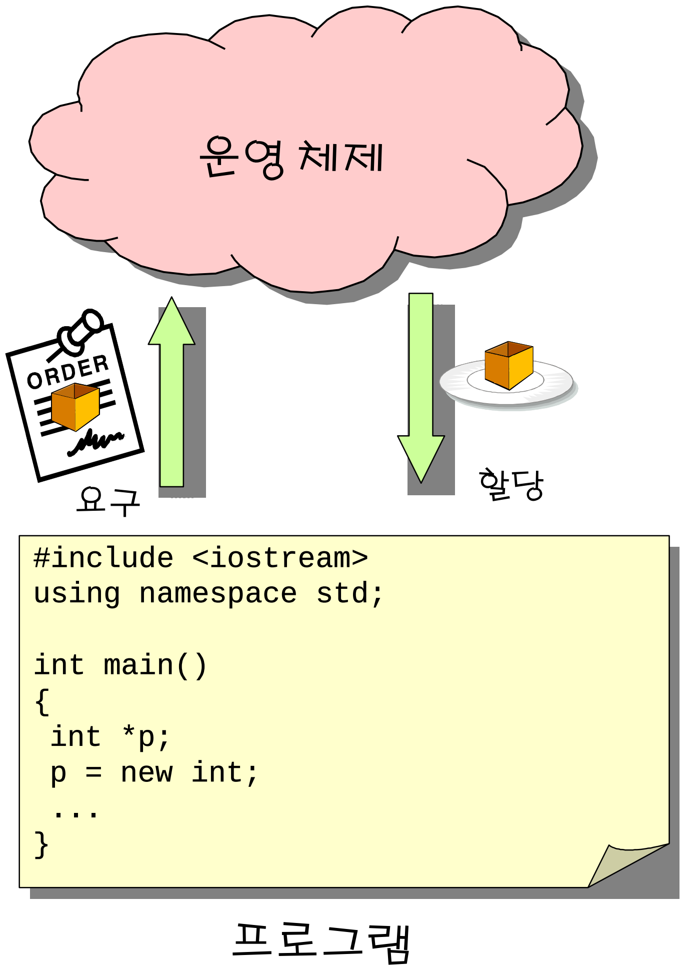
- 정적 메모리 할당
  - 프로그램이 시작되기 전에 미리 정해진 크기의 메모리를 할당받는 것
  - 메모리의 크기는 프로그램이 시작하기 전에 결정

```
int i, j;
int buffer[80];
char name[] = “data structure";
```

  - 처음에 결정된 크기보다 더 큰 입력이 들어온다면 처리하지 못함
  - 더 작은 입력이 들어온다면 남은 메모리 공간은 낭비

동적 메모리 할당

- 실행 도중에 동적으로 메모리를 할당받는 것
- 사용이 끝나면 시스템에 메모리를 반납
- 필요한 만큼만 할당을 받고 메모리를 효율적으로 사용
- new와 delete 키워드 사용



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

int main()
{
 int *p;
 p = new int;
 ...
}
```

동적 메모리 할당 방법

- in C++ new / delete
- in C malloc() / free()

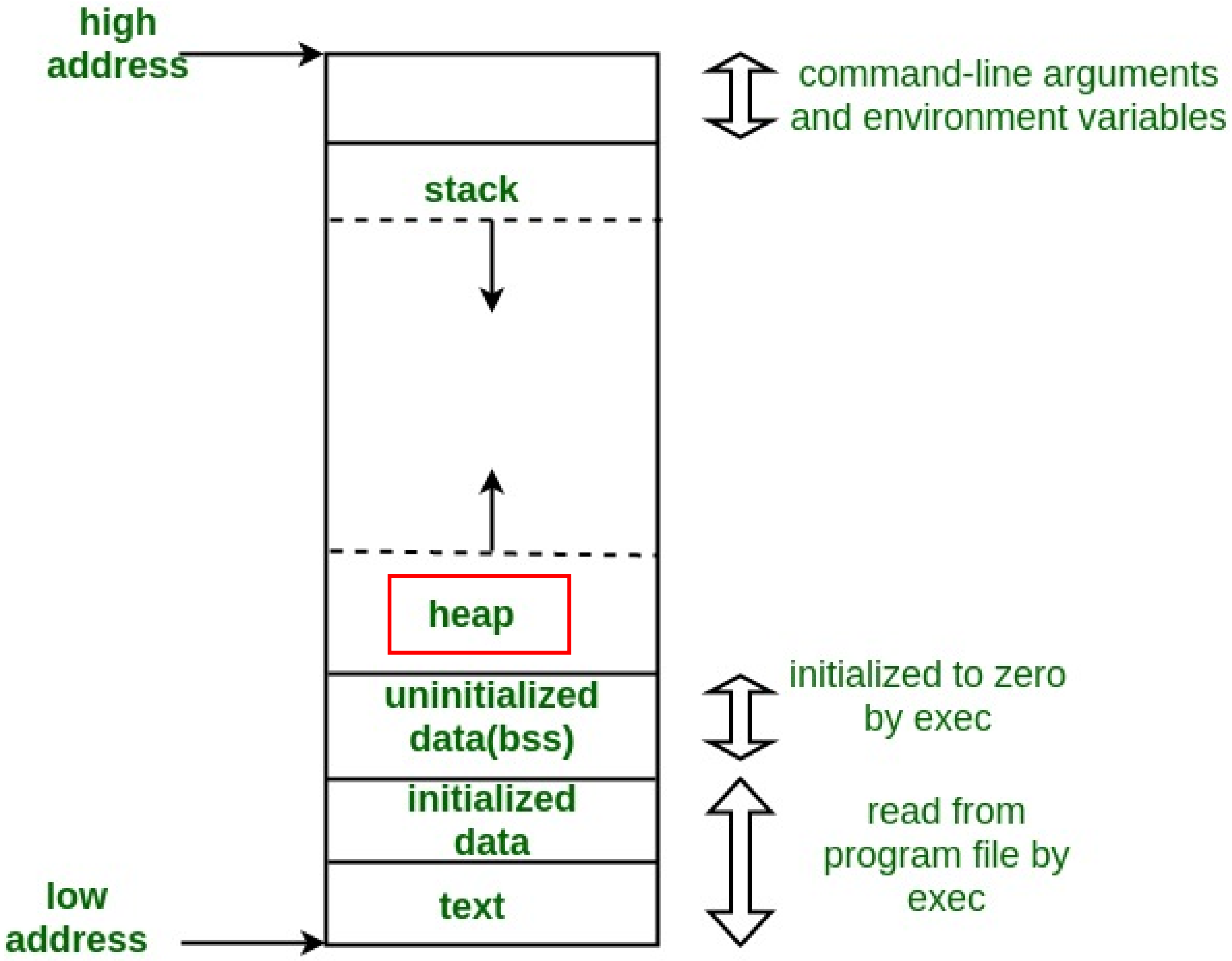
Typical Memory Layout of a C program

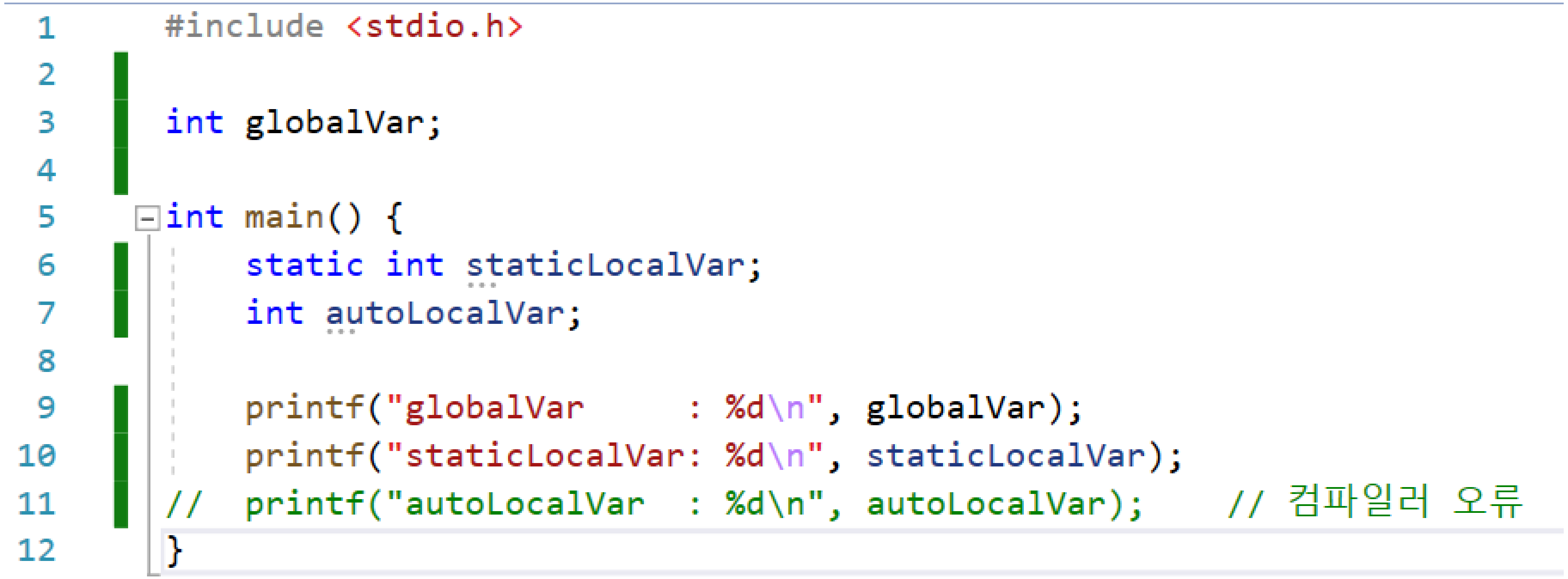
local variables

dynamically-allocated area

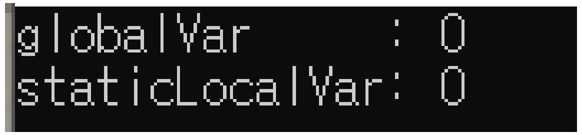
global, static variables

program code


high address
command-line arguments and environment variables
stack
heap
uninitialized data(bss)
initialized to zero by exec
initialized data
read from program file by exec
text
low address



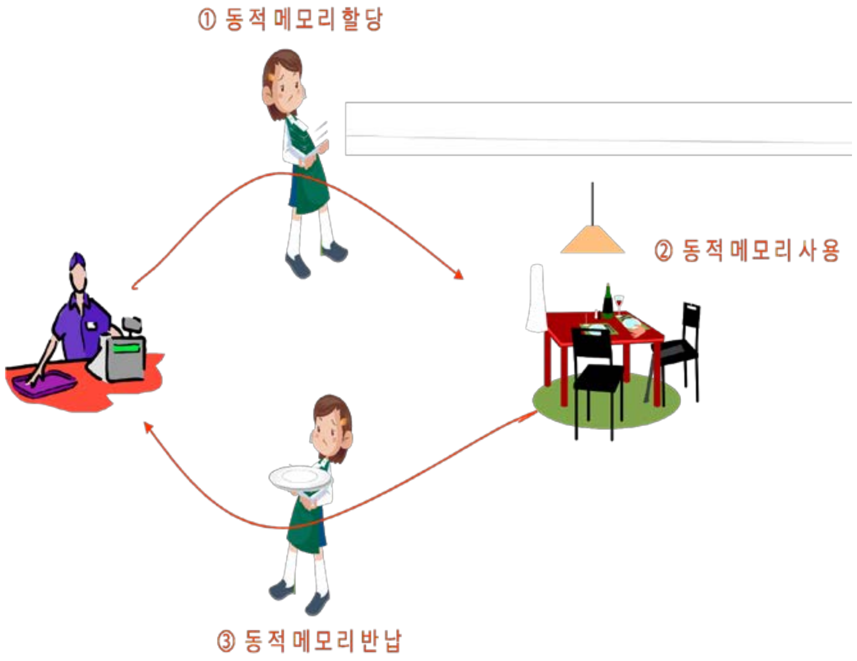
```c
#include <stdio.h>

int globalVar;

int main() {
    static int staticLocalVar;
    int autoLocalVar;

    printf("globalVar     : %d\n", globalVar);
    printf("staticLocalVar: %d\n", staticLocalVar);
//  printf("autoLocalVar  : %d\n", autoLocalVar);    // 컴파일러 오류
}
```



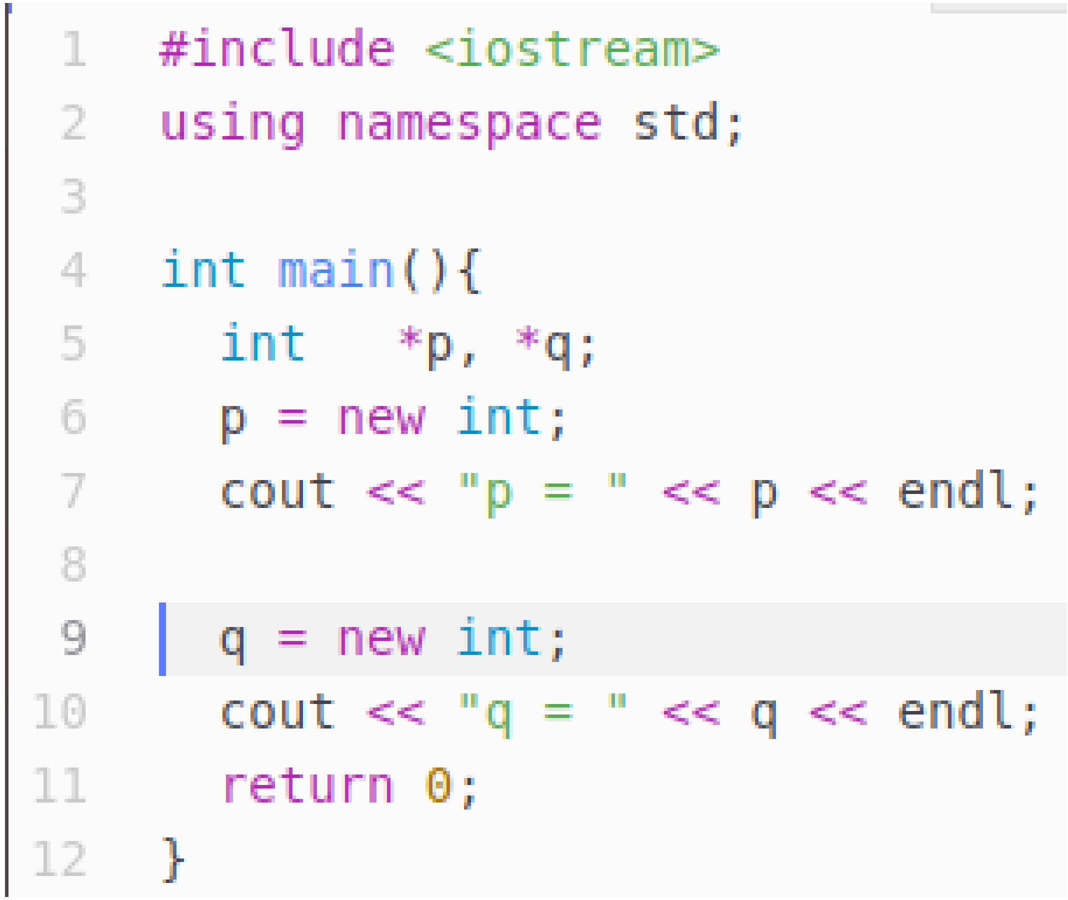
```
globalVar     : 0
staticLocalVar: 0
```

동적 메모리 할당의 과정

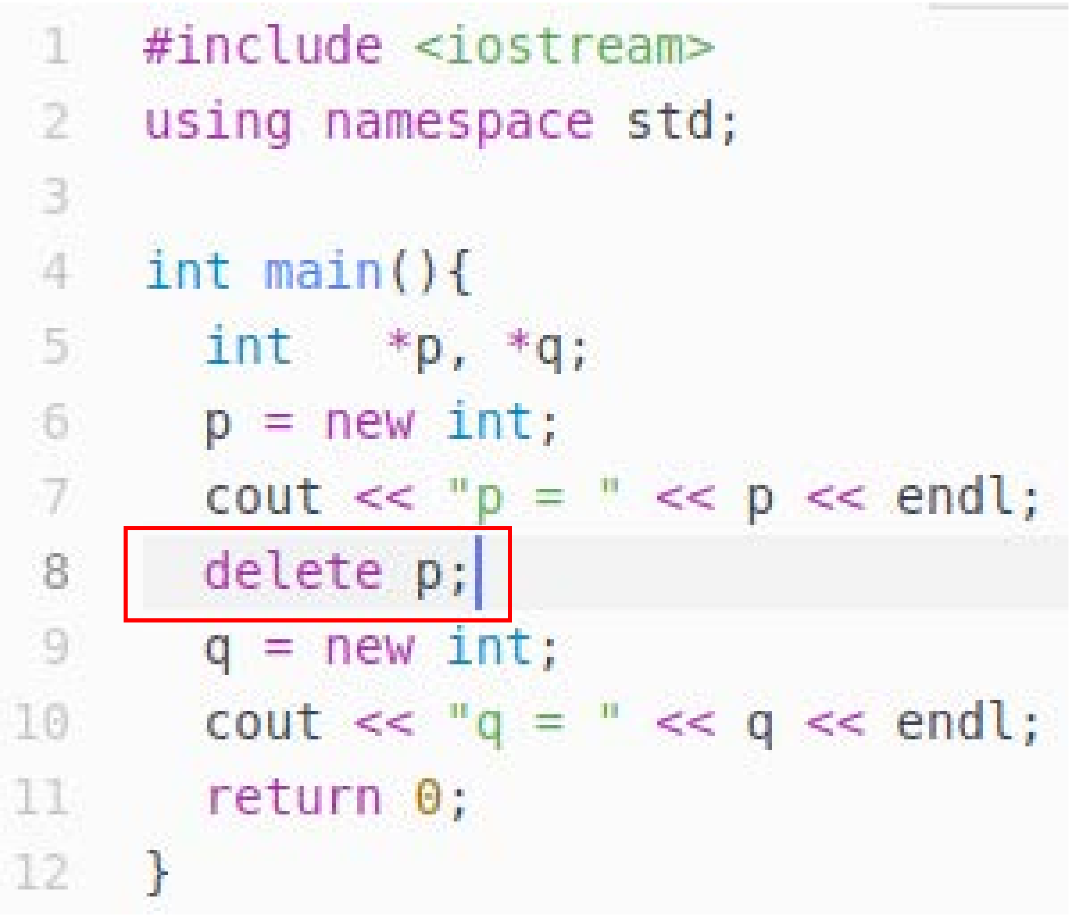
```
#include <iostream>
using namespace std;

int main()
{
        int *pi;     // 동적 메모리를 가리키는 포인터

        pi = new int;          // ① 동적 메모리 할당

        *pi = 100;             // ② 동적 메모리 사용
        delete pi;             // ③ 동적 메모리 반납

        int *q = new int;      // 반납된 메모리를 재사용할 수 있음
        return 0;
}
```



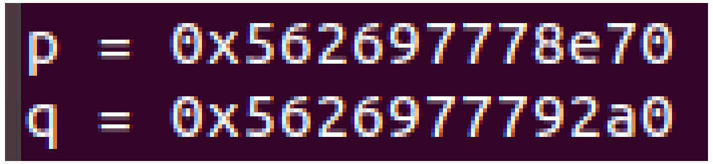
// pi == NULL 이면 delete 는 아무 것도 하지 않는다. error 가 아님



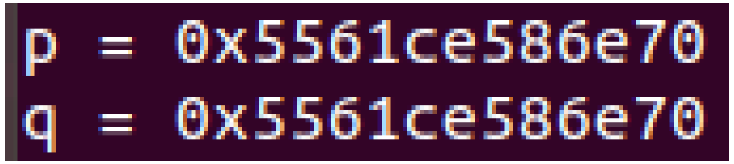
```cpp
#include <iostream>
using namespace std;

int main(){
  int   *p, *q;
  p = new int;
  cout << "p = " << p << endl;

  q = new int;
  cout << "q = " << q << endl;
  return 0;
}
```



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

int main(){
  int   *p, *q;
  p = new int;
  cout << "p = " << p << endl;
  delete p;
  q = new int;
  cout << "q = " << q << endl;
  return 0;
}
```



```
p = 0x562697778e70
q = 0x5626977792a0
```

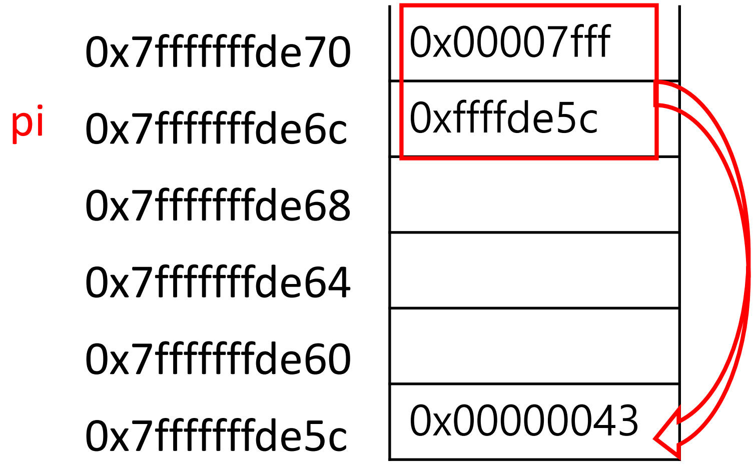


```
p = 0x5561ce586e70
q = 0x5561ce586e70
```

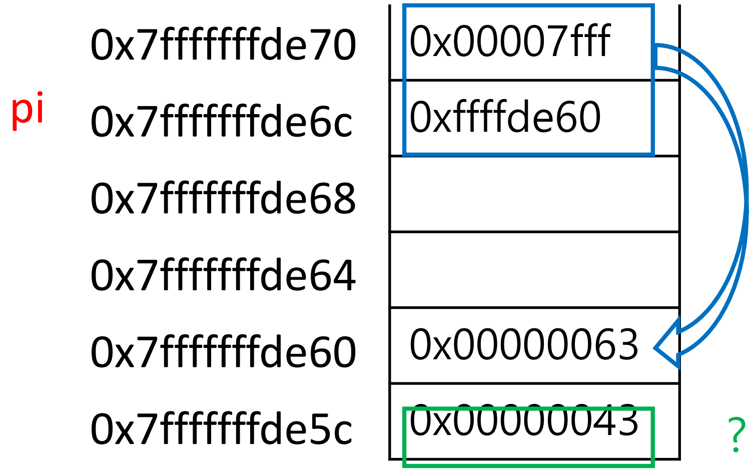
메모리 누수의 예제

```
void sub()
{
      int *p = new int;   // ①
      *p = 0x43;

      p = new int;        // ②
      *p = 0x63;
}
```

잘못된 버전



	0x7fffffffde70	0x00007fff
pi	0x7fffffffde6c	0xffffde5c
	0x7fffffffde68	
	0x7fffffffde64	
	0x7fffffffde60	
	0x7fffffffde5c	0x00000043



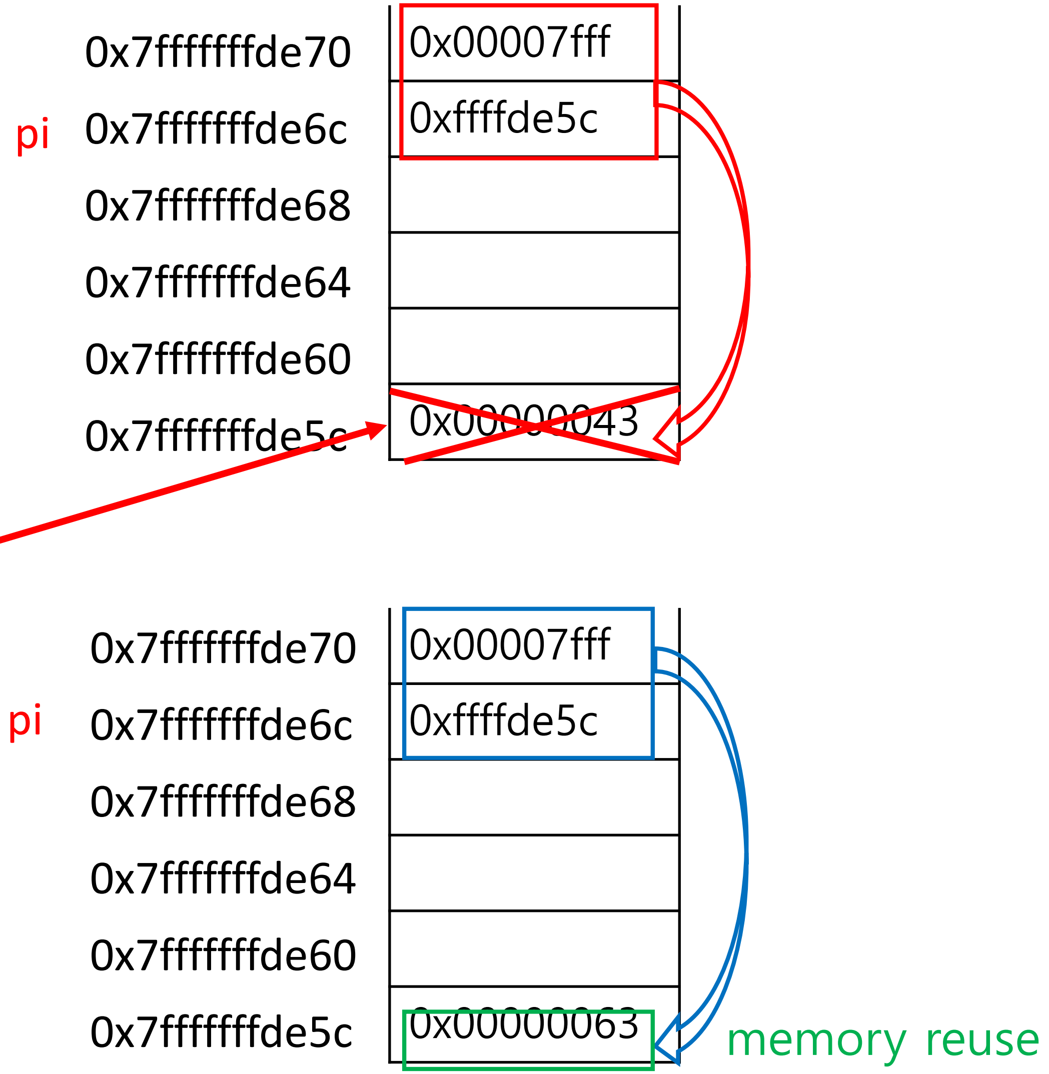
	0x7fffffffde70	0x00007fff
pi	0x7fffffffde6c	0xffffde60
	0x7fffffffde68	
	0x7fffffffde64	
	0x7fffffffde60	0x00000063
	0x7fffffffde5c	0x00000043

? memory leak

메모리 누수의 예제

```
void sub()
{
      int *p = new int;   // ①
      *p = 0x43;
      delete p;

      p = new int;         // ②
      *p = 0x63;
      delete p;
}
```

올바른 버전

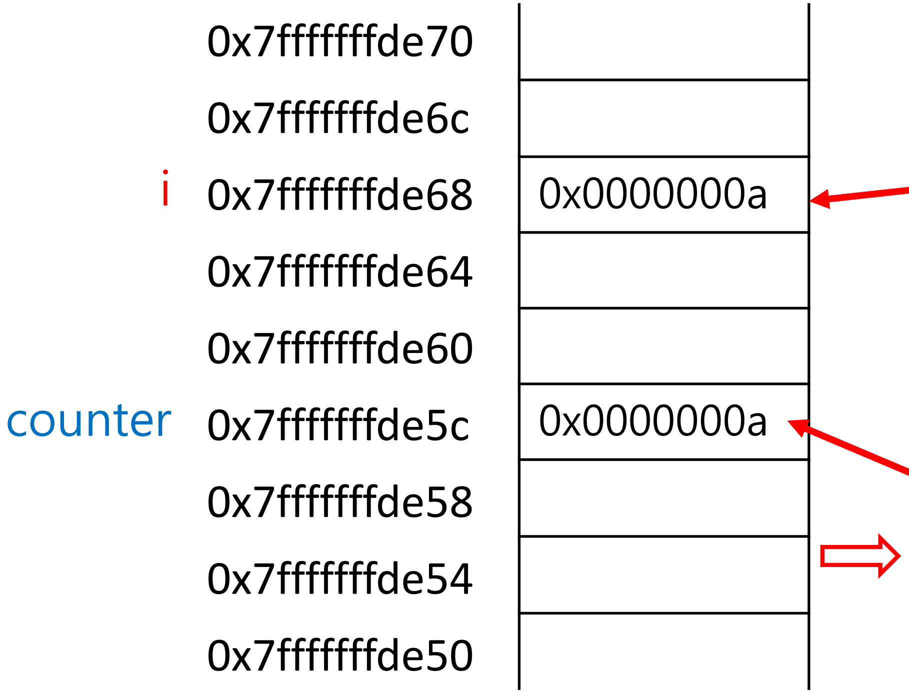


	0x7fffffffde70	0x00007fff
pi	0x7fffffffde6c	0xffffde5c
	0x7fffffffde68	
	0x7fffffffde64	
	0x7fffffffde60	
	0x7fffffffde5c	~~0x00000043~~

	0x7fffffffde70	0x00007fff
pi	0x7fffffffde6c	0xffffde5c
	0x7fffffffde68	
	0x7fffffffde64	
	0x7fffffffde60	
	0x7fffffffde5c	0x00000063

memory reuse

C 언어에서 함수의 인자 전달 방법은 call by value



	0x7fffffffde70	
	0x7fffffffde6c	
i	0x7fffffffde68	0x0000000a
	0x7fffffffde64	
	0x7fffffffde60	
counter	0x7fffffffde5c	0x0000000a
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	

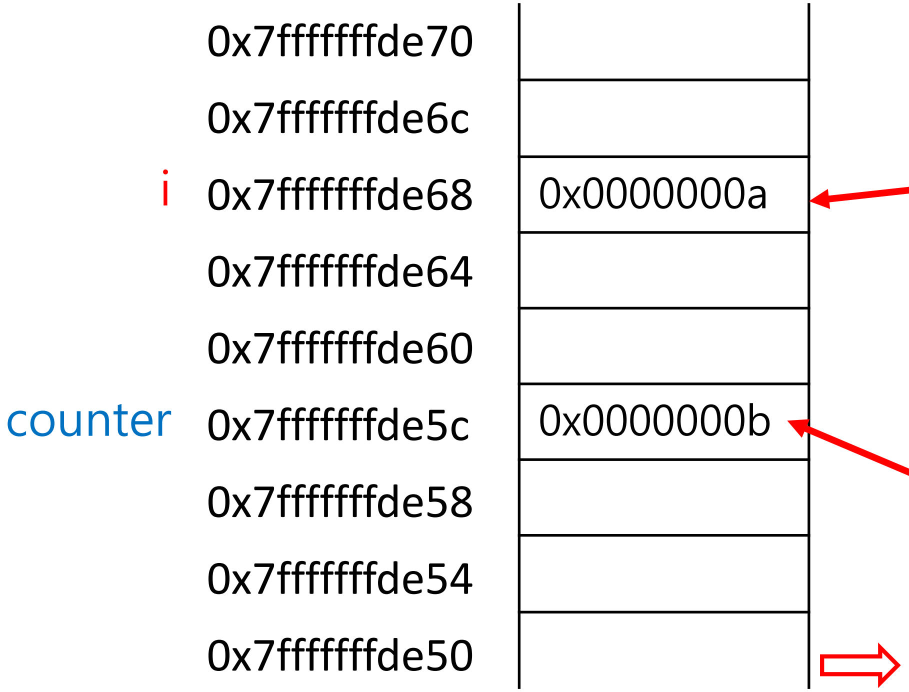
```
#include <iostream>
using namespace std;
int inc(int counter);
int main()
{
      int i;
      i = 10;
      cout << "함수 호출전 i=" << i << endl;
      inc(i);
      cout << "함수 호출후 i=" << i << endl;
      return 0;
}
int inc(int counter)
{
      counter++;
      return counter;
}
```

값에 의한 호출
(call by value)

매개 변수도 일종의 지역 변수임

```
함수 호출전 i=10
```

C 언어에서 함수의 인자 전달 방법은 call by value



	0x7fffffffde70	
	0x7fffffffde6c	
i	0x7fffffffde68	0x0000000a
	0x7fffffffde64	
	0x7fffffffde60	
counter	0x7fffffffde5c	0x0000000b
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	

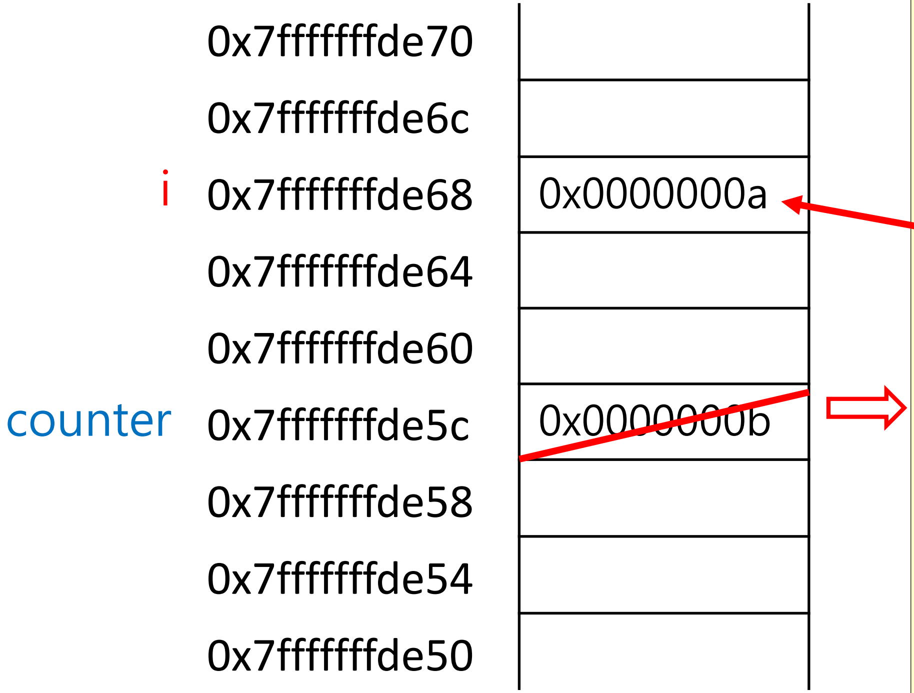
```
#include <iostream>
using namespace std;
int inc(int counter);
int main()
{
      int i;
      i = 10;
      cout << "함수 호출전 i=" << i << endl;
      inc(i);
      cout << "함수 호출후 i=" << i << endl;
      return 0;
}
int inc(int counter)
{
      counter++;
      return counter;
}
```

값에 의한 호출
(call by value)

매개 변수도 일종의 지역 변수임

```
함수 호출전 i=10
```

C 언어에서 함수의 인자 전달 방법은 call by value



	0x7fffffffde70	
	0x7fffffffde6c	
i	0x7fffffffde68	0x0000000a
	0x7fffffffde64	
	0x7fffffffde60	
counter	0x7fffffffde5c	~~0x0000000b~~
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	

```
#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;
}
```

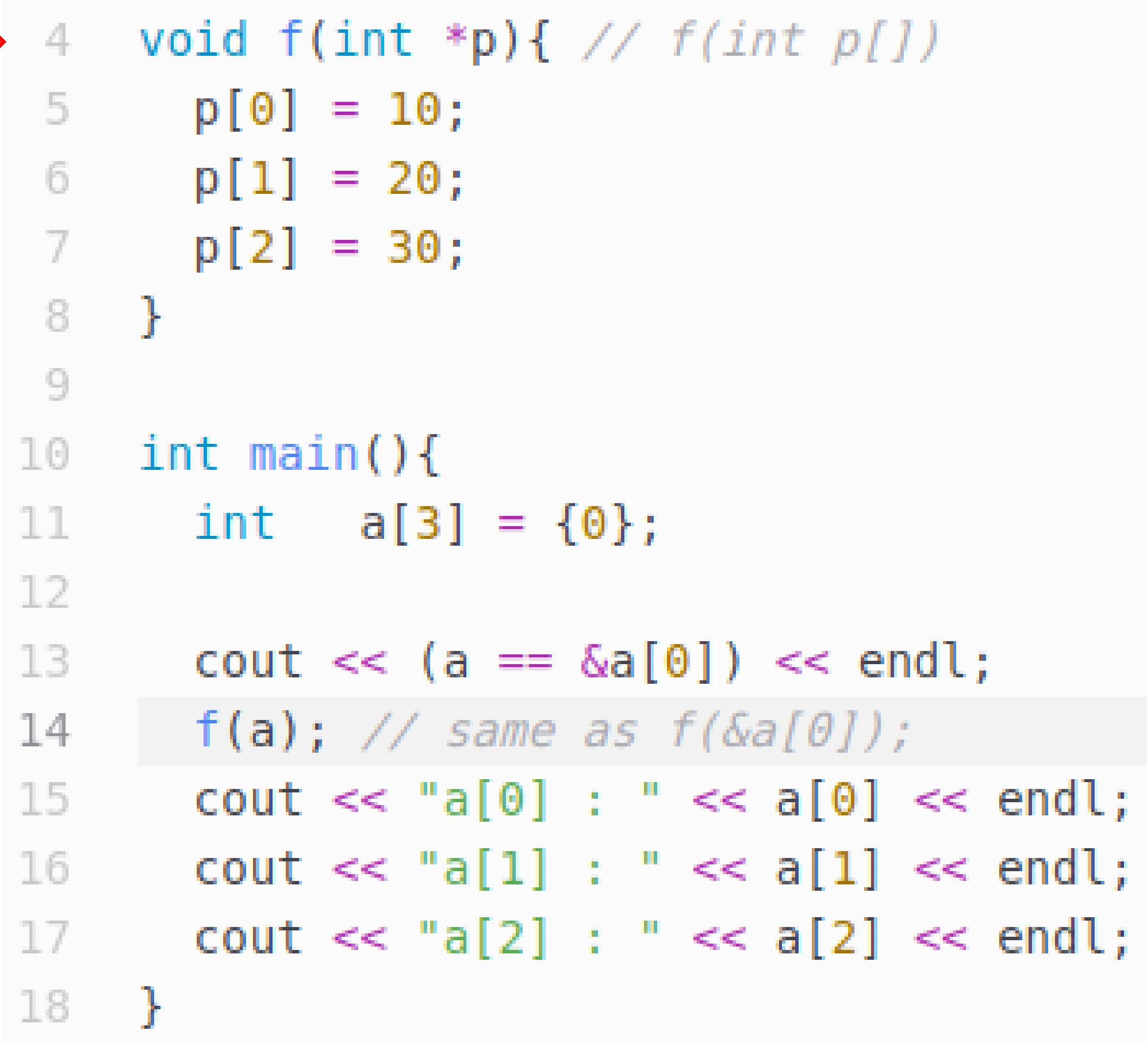
값에 의한 호출
(call by value)

매개 변수도 일종의 지역 변수임

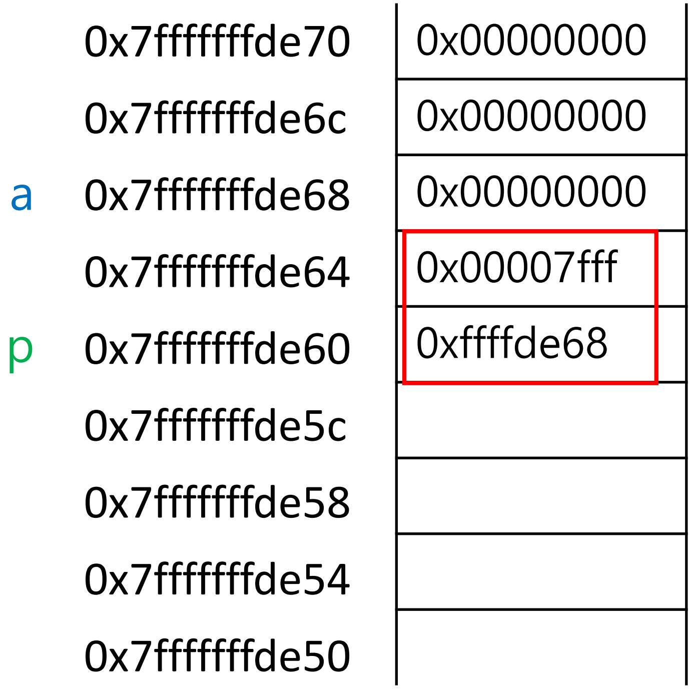
```
함수 호출전 i=10
함수 호출후 i=10
```

함수 인자로 배열 전달하기

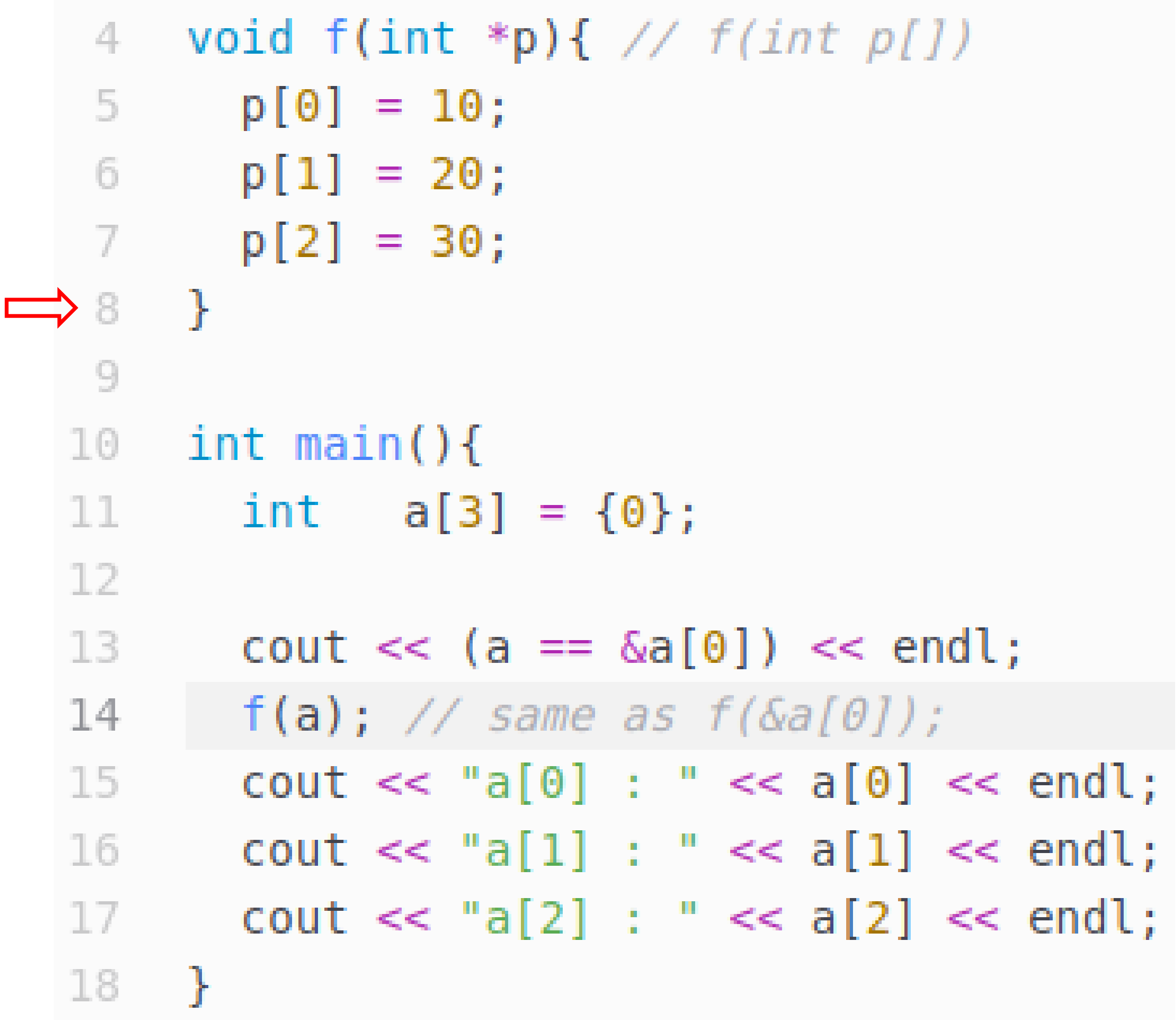
- 실행문에서 배열의 이름만 쓰면 배열의 시작 주소를 의미한다.



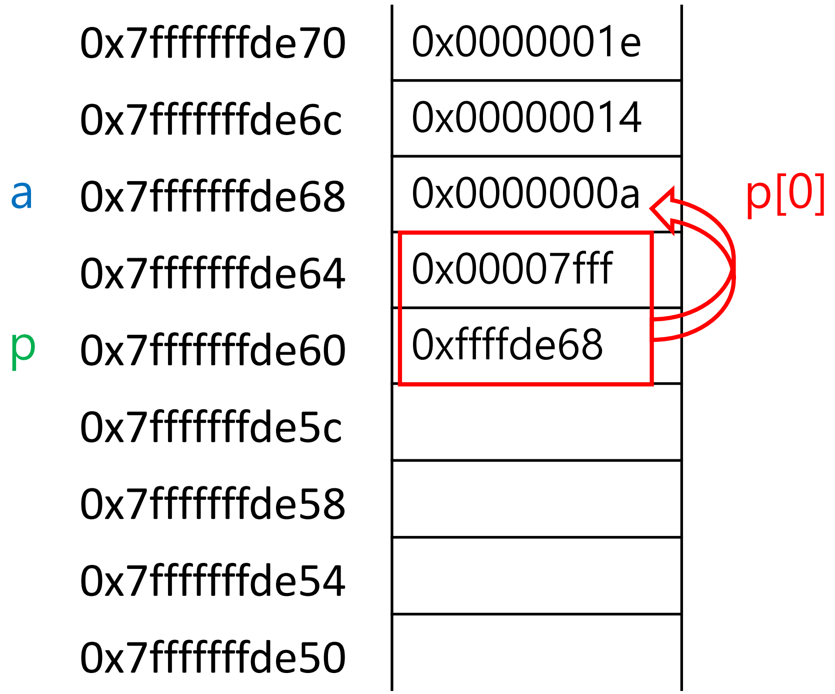
```cpp
void f(int *p){ // f(int p[])
  p[0] = 10;
  p[1] = 20;
  p[2] = 30;
}

int main(){
  int    a[3] = {0};

  cout << (a == &a[0]) << endl;
  f(a); // same as f(&a[0]);
  cout << "a[0] : " << a[0] << endl;
  cout << "a[1] : " << a[1] << endl;
  cout << "a[2] : " << a[2] << endl;
}
```



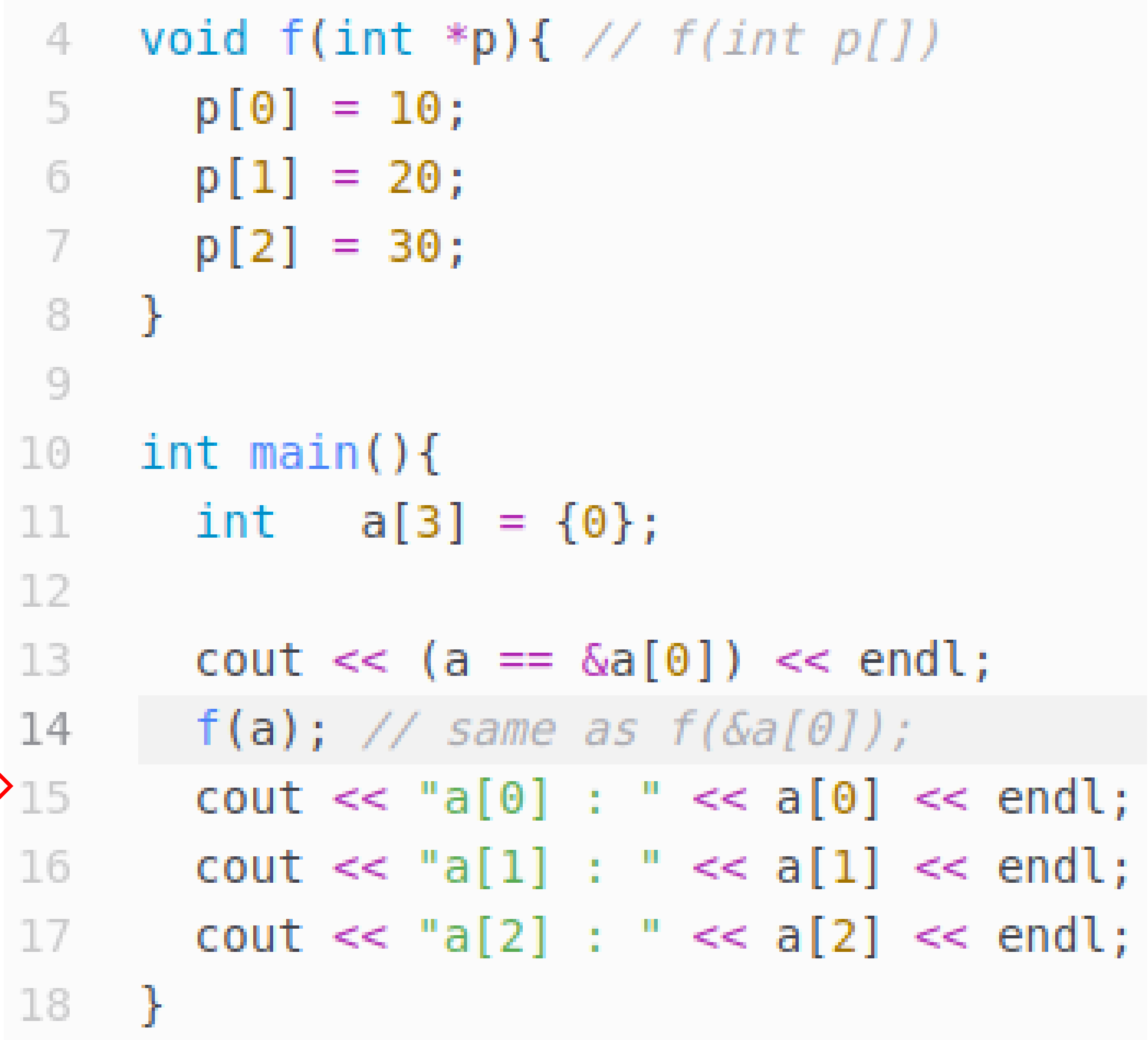
	0x7fffffffde70	0x00000000
	0x7fffffffde6c	0x00000000
a	0x7fffffffde68	0x00000000
	0x7fffffffde64	0x00007fff
p	0x7fffffffde60	0xffffde68
	0x7fffffffde5c	
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	



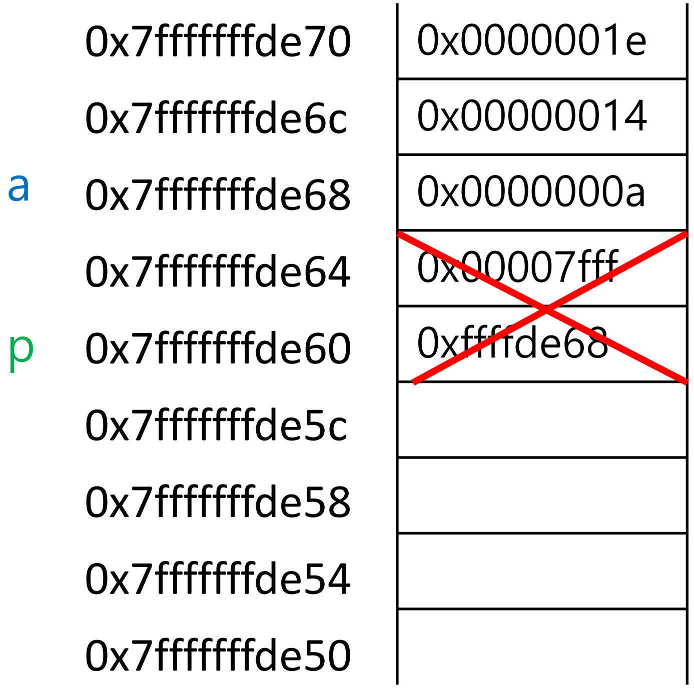
```cpp
void f(int *p){ // f(int p[])
  p[0] = 10;
  p[1] = 20;
  p[2] = 30;
}

int main(){
  int   a[3] = {0};

  cout << (a == &a[0]) << endl;
  f(a); // same as f(&a[0]);
  cout << "a[0] : " << a[0] << endl;
  cout << "a[1] : " << a[1] << endl;
  cout << "a[2] : " << a[2] << endl;
}
```



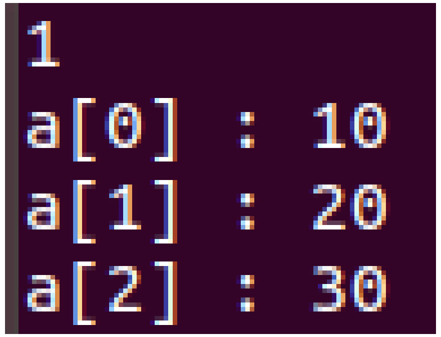
	0x7fffffffde70	0x0000001e
	0x7fffffffde6c	0x00000014
a	0x7fffffffde68	0x0000000a
	0x7fffffffde64	0x00007fff
p	0x7fffffffde60	0xffffde68
	0x7fffffffde5c	
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	



```cpp
void f(int *p){ // f(int p[])
  p[0] = 10;
  p[1] = 20;
  p[2] = 30;
}

int main(){
  int   a[3] = {0};

  cout << (a == &a[0]) << endl;
  f(a); // same as f(&a[0]);
  cout << "a[0] : " << a[0] << endl;
  cout << "a[1] : " << a[1] << endl;
  cout << "a[2] : " << a[2] << endl;
}
```

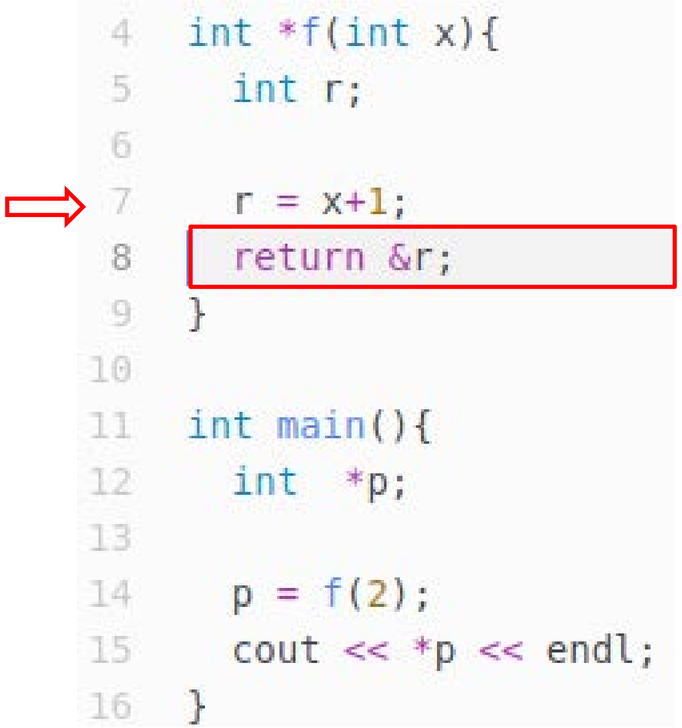


	0x7fffffffde70	0x0000001e
	0x7fffffffde6c	0x00000014
a	0x7fffffffde68	0x0000000a
	0x7fffffffde64	~~0x00007fff~~
p	0x7fffffffde60	~~0xffffde68~~
	0x7fffffffde5c	
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	



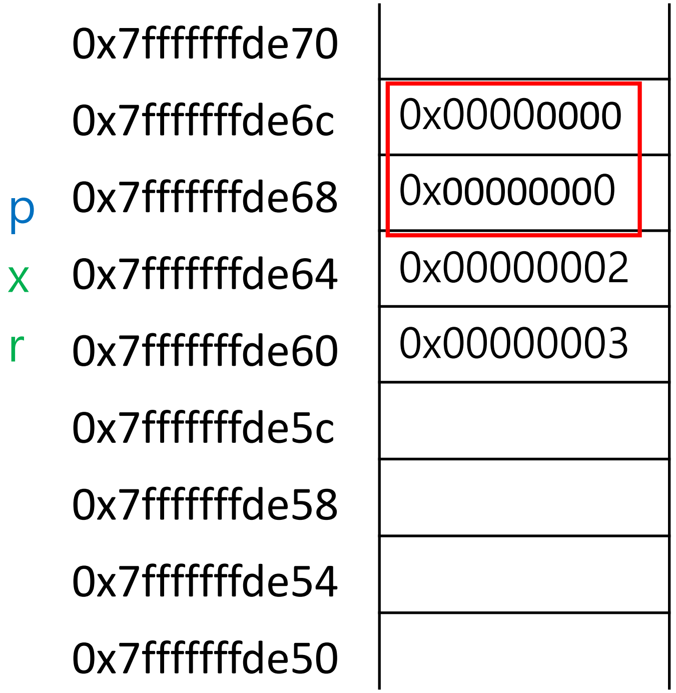
```
1
a[0] : 10
a[1] : 20
a[2] : 30
```

주의!!!

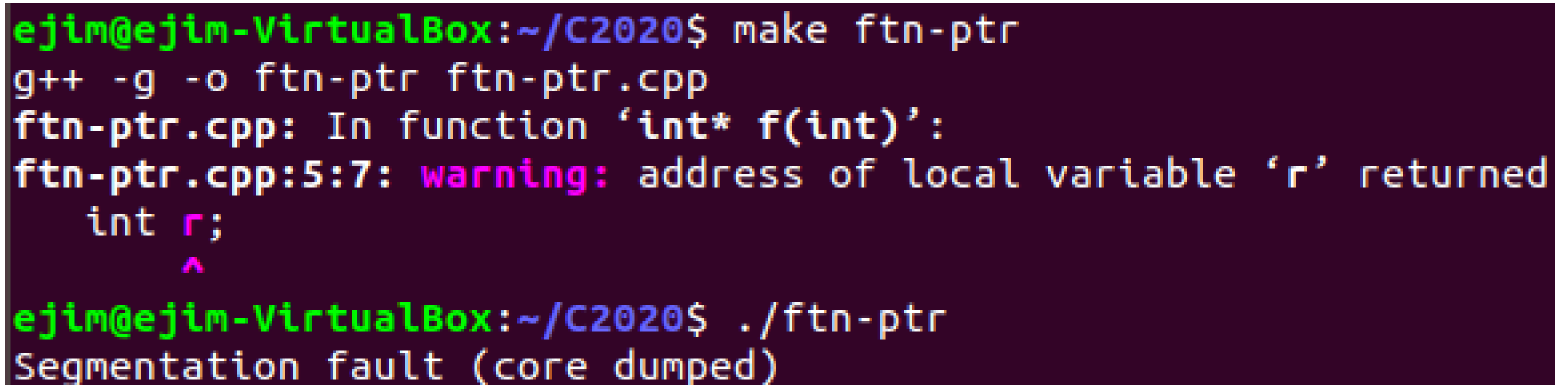


```
int *f(int x){
  int r;

  r = x+1;
  return &r;
}

int main(){
  int  *p;

  p = f(2);
  cout << *p << endl;
}
```

지역 변수 r는 함수가 종료되면 소멸되므로 그 주소를 반환하면 안된다.!!

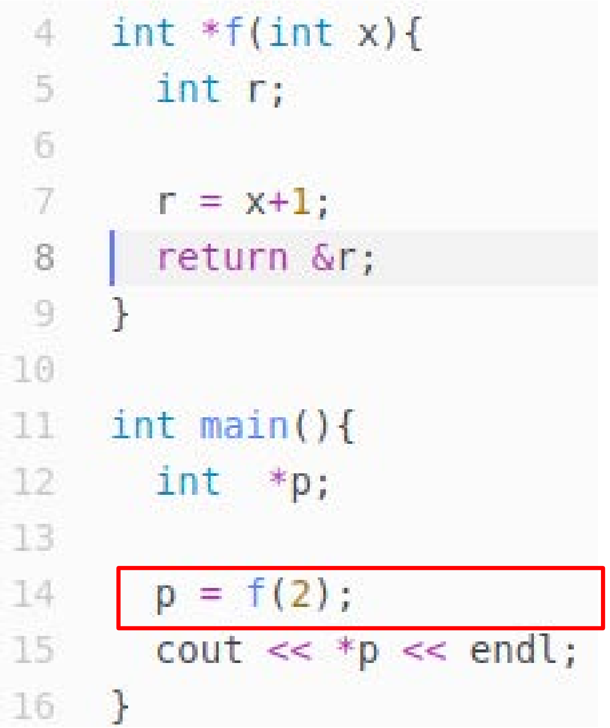


	0x7fffffffde70	
	0x7fffffffde6c	0x00000000
p	0x7fffffffde68	0x00000000
x	0x7fffffffde64	0x00000002
r	0x7fffffffde60	0x00000003
	0x7fffffffde5c	
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	

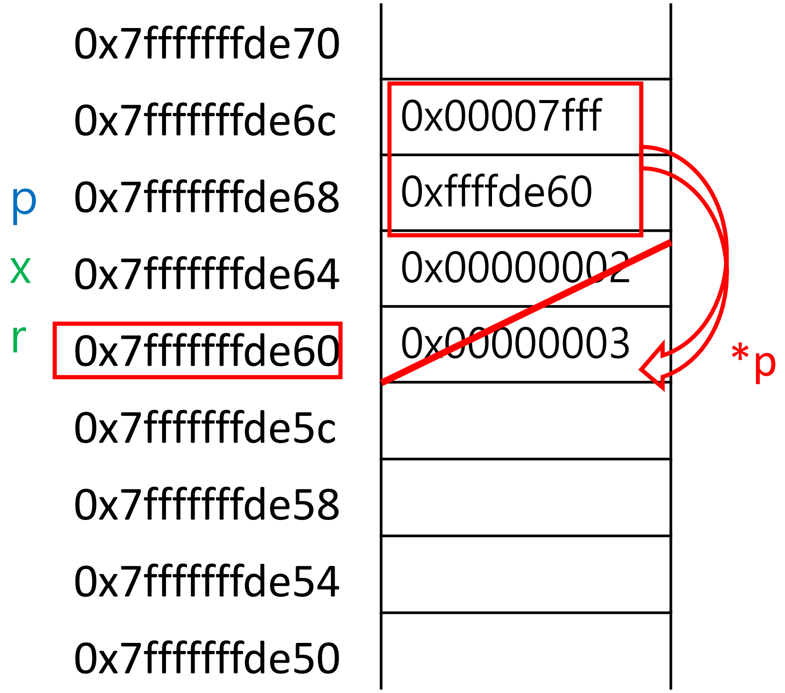


```
ejim@ejim-VirtualBox:~/C2020$ make ftn-ptr
g++ -g -o ftn-ptr ftn-ptr.cpp
ftn-ptr.cpp: In function ‘int* f(int)’:
ftn-ptr.cpp:5:7: warning: address of local variable ‘r’ returned
   int r;
       ^
ejim@ejim-VirtualBox:~/C2020$ ./ftn-ptr
Segmentation fault (core dumped)
```

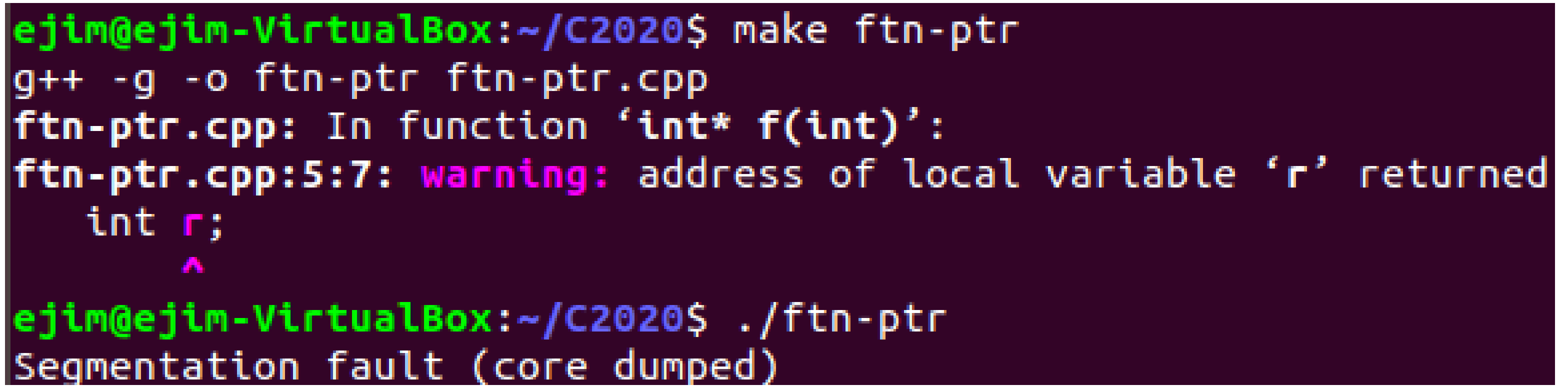
지역 변수의 주소를 반환하면 , 함수가 종료되면 사라지기 때문에 오류



```
int *f(int x){
  int r;

  r = x+1;
  return &r;
}

int main(){
  int  *p;

  p = f(2);
  cout << *p << endl;
}
```

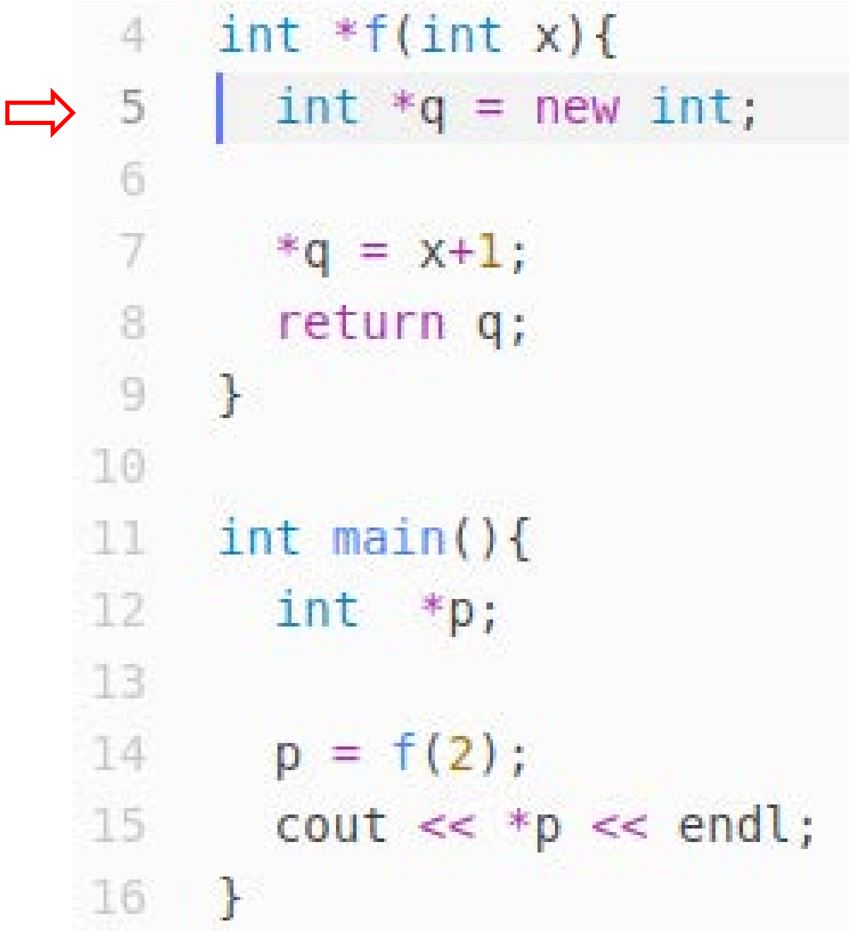


	0x7fffffffde70	
	0x7fffffffde6c	0x00007fff
p	0x7fffffffde68	0xffffde60
x	0x7fffffffde64	~~0x00000002~~
r	0x7fffffffde60	~~0x00000003~~
	0x7fffffffde5c	
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	

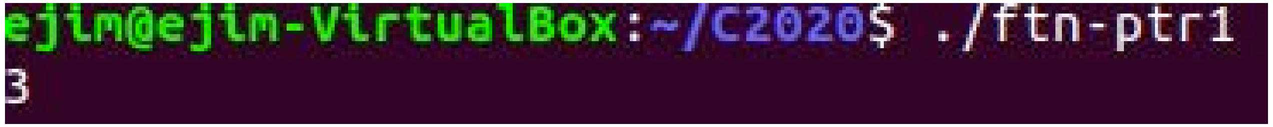


```
ejim@ejim-VirtualBox:~/C2020$ make ftn-ptr
g++ -g -o ftn-ptr ftn-ptr.cpp
ftn-ptr.cpp: In function ‘int* f(int)’:
ftn-ptr.cpp:5:7: warning: address of local variable ‘r’ returned
   int r;
       ^
ejim@ejim-VirtualBox:~/C2020$ ./ftn-ptr
Segmentation fault (core dumped)
```

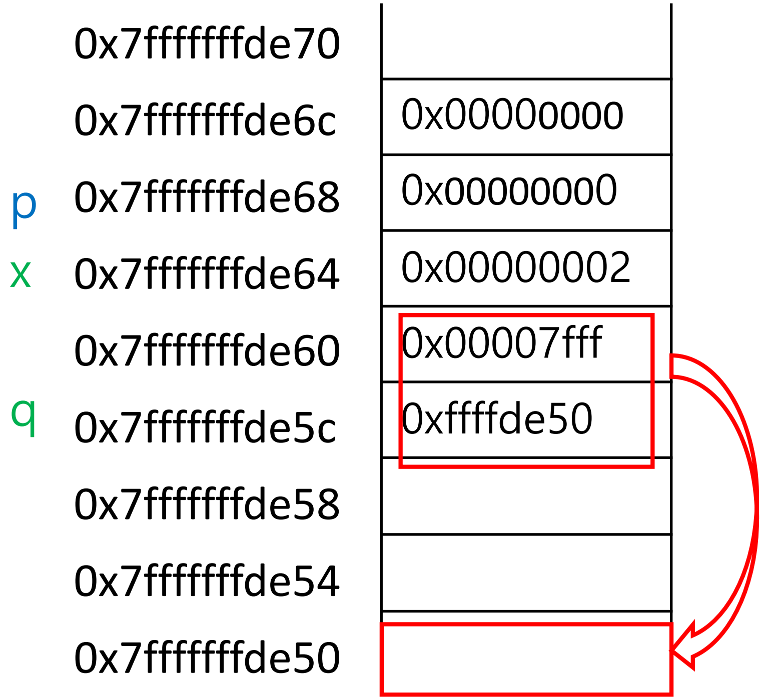
그러나 동적 할당받은 주소를 반환하는 것은 가능하다.



```
int *f(int x){
  int *q = new int;

  *q = x+1;
  return q;
}

int main(){
  int  *p;

  p = f(2);
  cout << *p << endl;
}
```

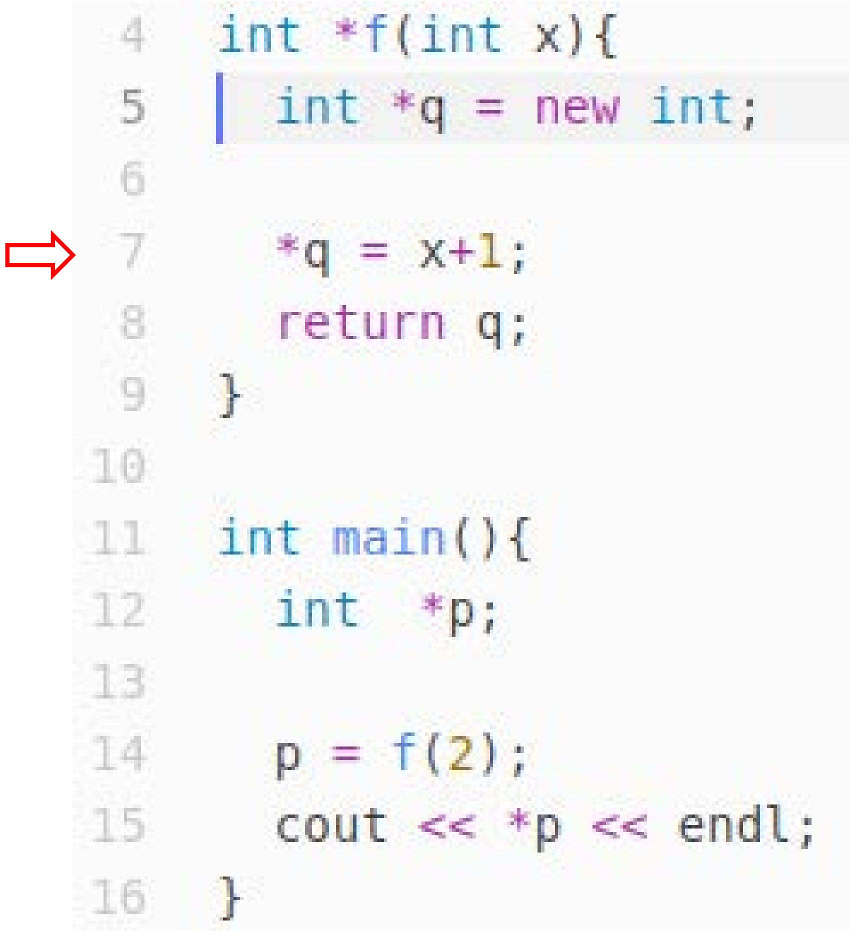


```
ejim@ejim-VirtualBox:~/C2020$ ./ftn-ptr1
3
```

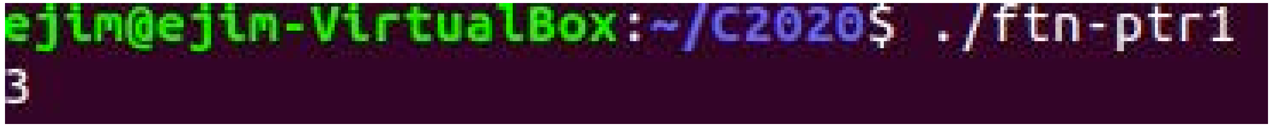


	0x7fffffffde70	
	0x7fffffffde6c	0x00000000
p	0x7fffffffde68	0x00000000
x	0x7fffffffde64	0x00000002
	0x7fffffffde60	0x00007fff
q	0x7fffffffde5c	0xffffde50
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	

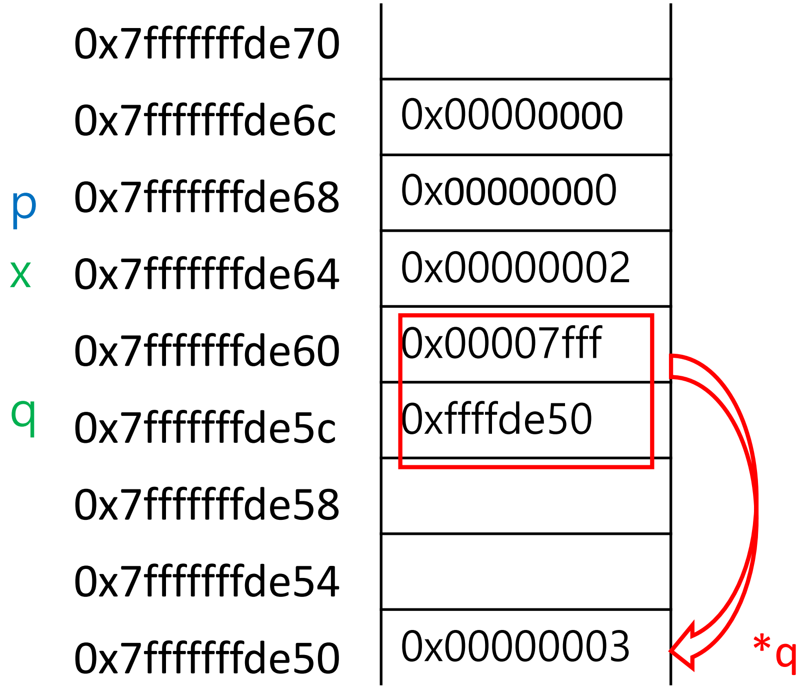
그러나 동적 할당받은 주소를 반환하는 것은 가능하다.



```
int *f(int x){
  int *q = new int;

  *q = x+1;
  return q;
}

int main(){
  int  *p;

  p = f(2);
  cout << *p << endl;
}
```

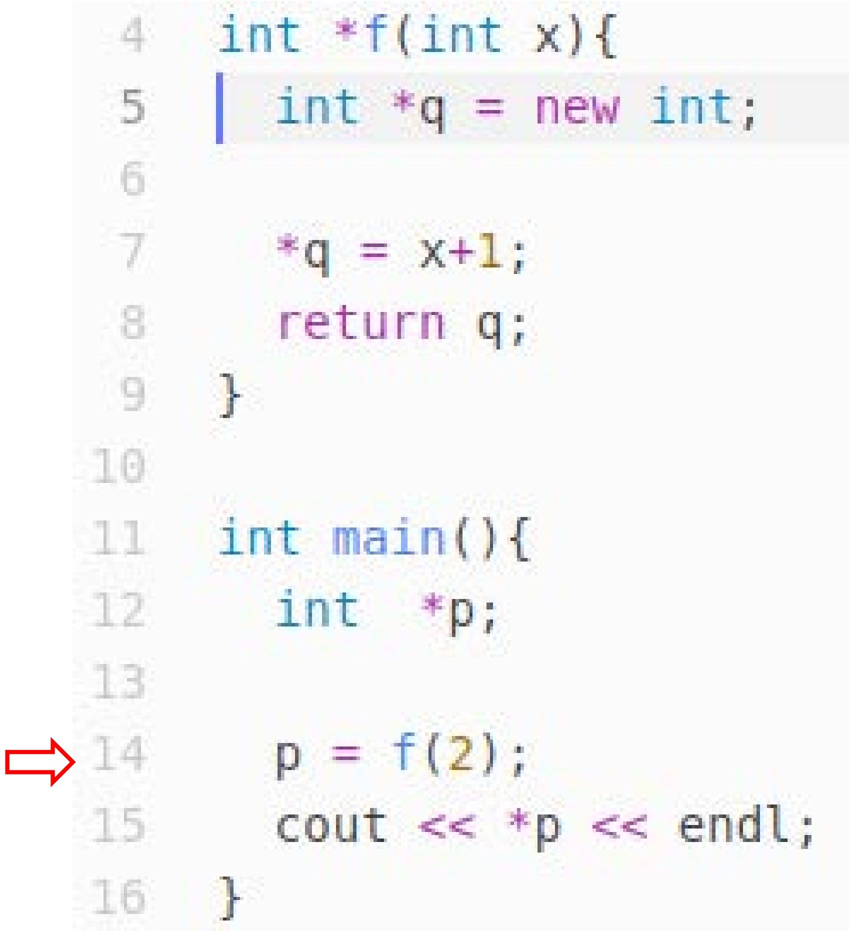


```
ejim@ejim-VirtualBox:~/C2020$ ./ftn-ptr1
3
```

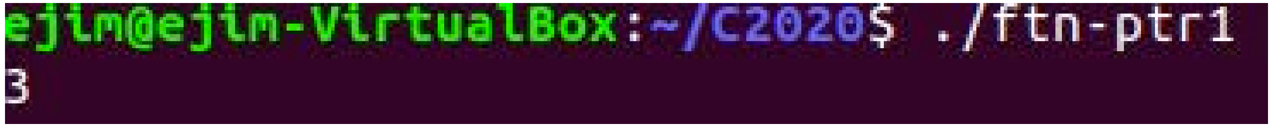


	0x7fffffffde70		
	0x7fffffffde6c	0x00000000	
p	0x7fffffffde68	0x00000000	
x	0x7fffffffde64	0x00000002	
	0x7fffffffde60	0x00007fff	
q	0x7fffffffde5c	0xffffde50	
	0x7fffffffde58		
	0x7fffffffde54		
	0x7fffffffde50	0x00000003	*q

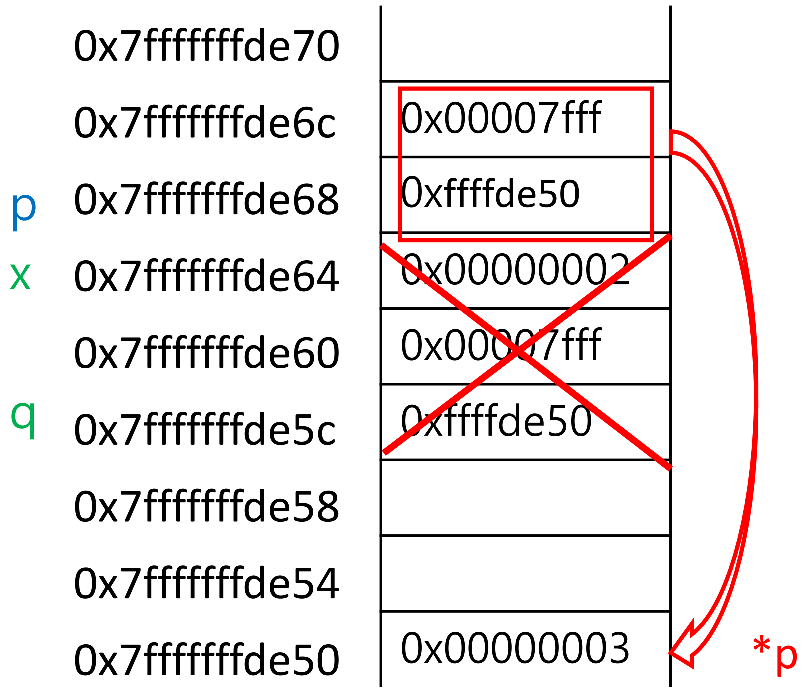
그러나 동적 할당받은 주소를 반환하는 것은 가능하다.



```
int *f(int x){
  int *q = new int;

  *q = x+1;
  return q;
}

int main(){
  int  *p;

  p = f(2);
  cout << *p << endl;
}
```



```
ejim@ejim-VirtualBox:~/C2020$ ./ftn-ptr1
3
```



	0x7fffffffde70	
	0x7fffffffde6c	0x00007fff
p	0x7fffffffde68	0xffffde50
x	0x7fffffffde64	0x00000002
	0x7fffffffde60	0x00007fff
q	0x7fffffffde5c	0xffffde50
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	0x00000003

1차원 배열을 동적으로 할당

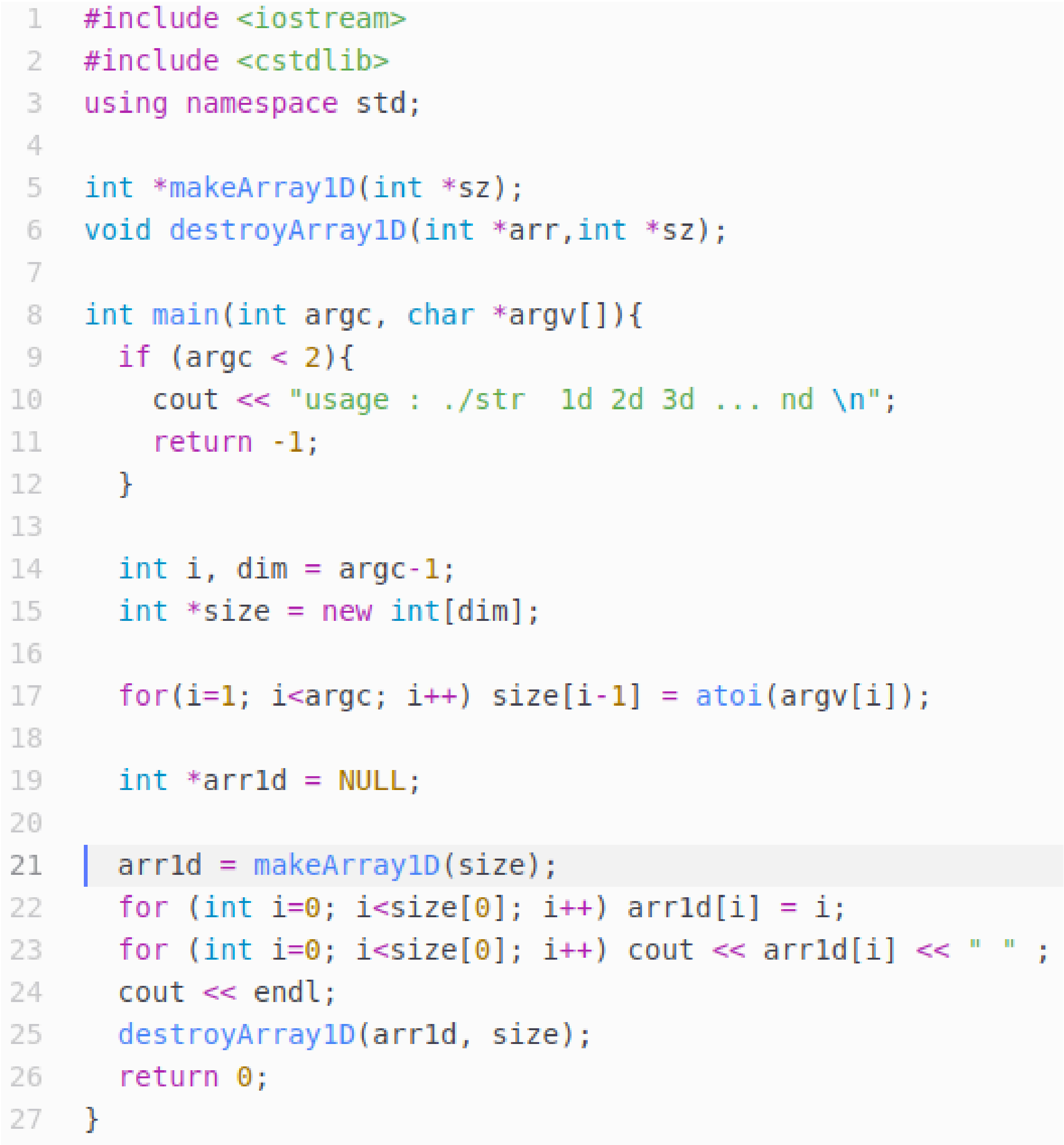
- 배열

```
double *pd = new double[10];
…
delete[] pd;
```

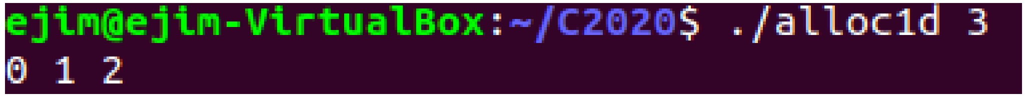
실습

- 3차원 배열을 만들어서 반환하는 함수 makeArray3D() 와 3차원 함수를 heap 에서 제거하는 함수 destroyArray3D() 를 완성하라.

1차원 배열 만들어 반환하기 makeArray1D()



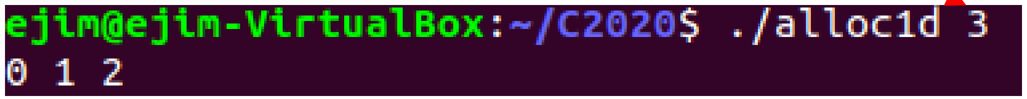
```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int *makeArray1D(int *sz);
void destroyArray1D(int *arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int *arr1d = NULL;

  arr1d = makeArray1D(size);
  for (int i=0; i<size[0]; i++) arr1d[i] = i;
  for (int i=0; i<size[0]; i++) cout << arr1d[i] << " " ;
  cout << endl;
  destroyArray1D(arr1d, size);
  return 0;
}
```



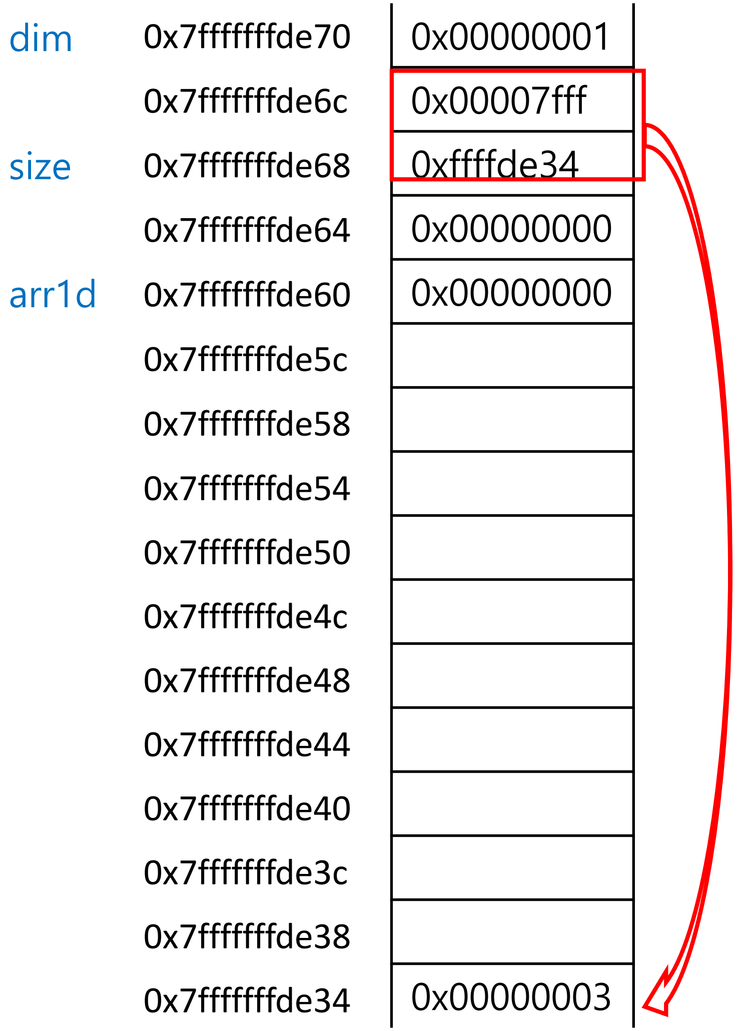
```
ejim@ejim-VirtualBox:~/C2020$ ./alloc1d 3
0 1 2
```

1차원 배열 만들어 반환하기 makeArray1D()

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

int *makeArray1D(int *sz);
void destroyArray1D(int *arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int *arr1d = NULL;

  arr1d = makeArray1D(size);
  for (int i=0; i<size[0]; i++) arr1d[i] = i;
  for (int i=0; i<size[0]; i++) cout << arr1d[i] << " " ;
  cout << endl;
  destroyArray1D(arr1d, size);
  return 0;
}
```

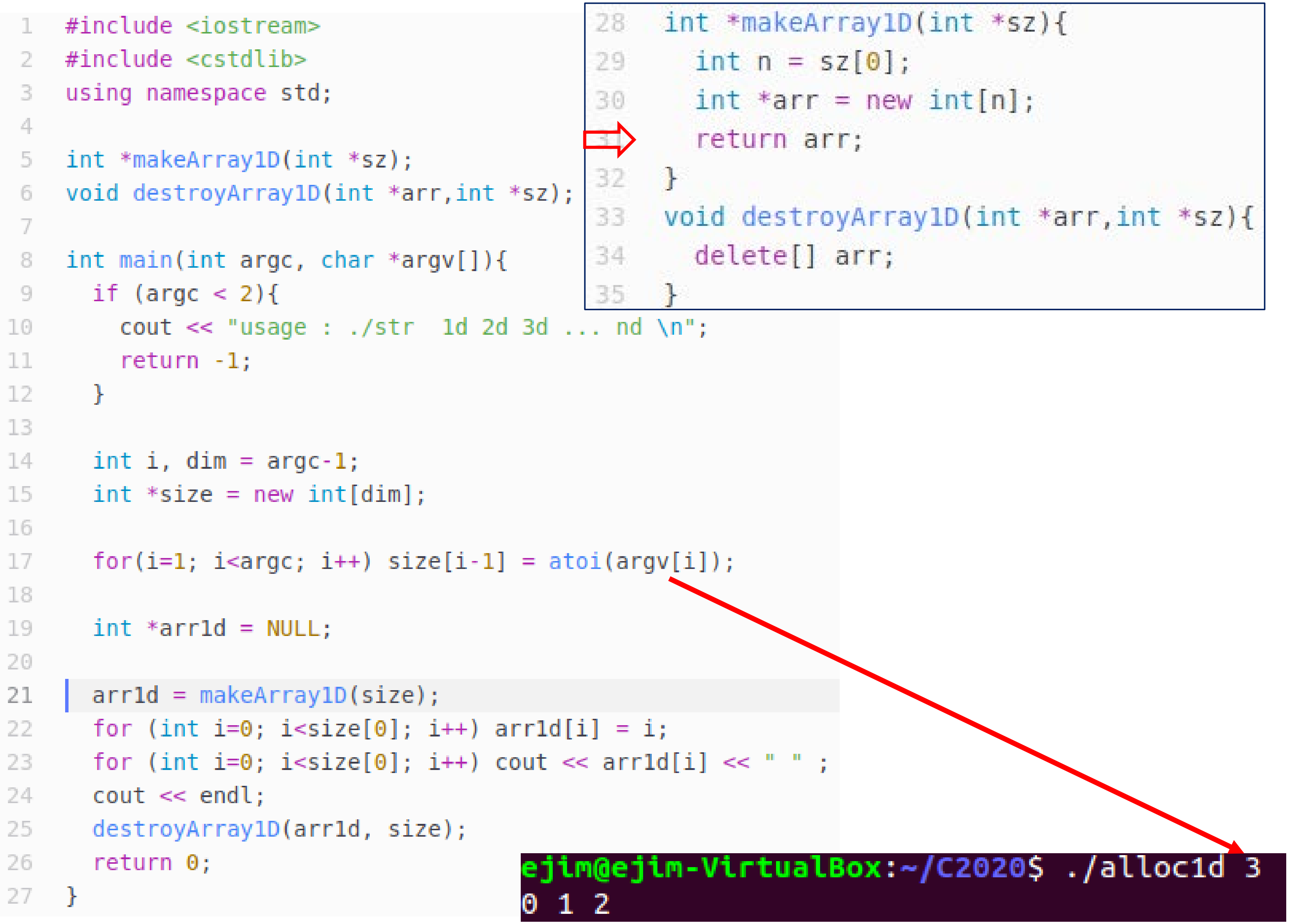


```
ejim@ejim-VirtualBox:~/C2020$ ./alloc1d 3
0 1 2
```



dim	0x7fffffffde70	0x00000001
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde34
	0x7fffffffde64	0x00000000
arr1d	0x7fffffffde60	0x00000000
	0x7fffffffde5c	
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	
	0x7fffffffde4c	
	0x7fffffffde48	
	0x7fffffffde44	
	0x7fffffffde40	
	0x7fffffffde3c	
	0x7fffffffde38	
	0x7fffffffde34	0x00000003

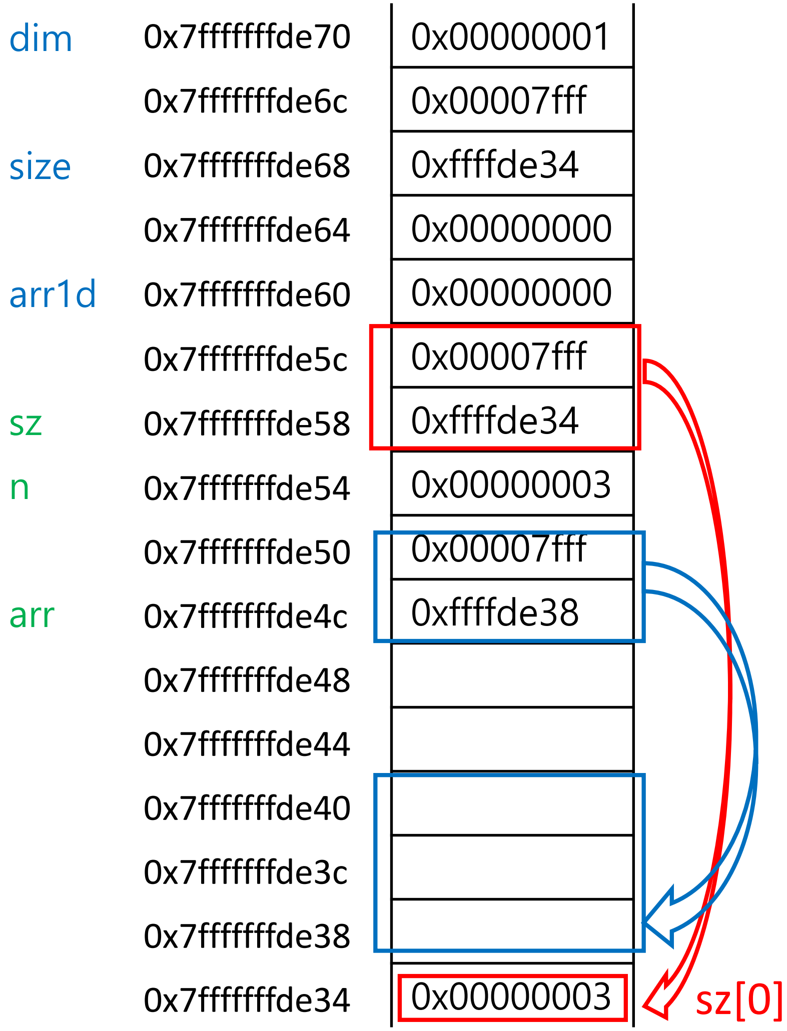
1차원 배열 만들어 반환하기 makeArray1D()



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

int *makeArray1D(int *sz);
void destroyArray1D(int *arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int *arr1d = NULL;

  arr1d = makeArray1D(size);
  for (int i=0; i<size[0]; i++) arr1d[i] = i;
  for (int i=0; i<size[0]; i++) cout << arr1d[i] << " " ;
  cout << endl;
  destroyArray1D(arr1d, size);
  return 0;
}
```

```cpp
int *makeArray1D(int *sz){
  int n = sz[0];
  int *arr = new int[n];
  return arr;
}
void destroyArray1D(int *arr,int *sz){
  delete[] arr;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./alloc1d 3
0 1 2
```

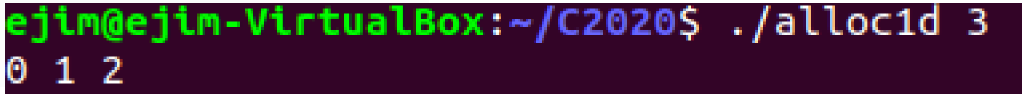


dim	0x7fffffffde70	0x00000001
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde34
	0x7fffffffde64	0x00000000
arr1d	0x7fffffffde60	0x00000000
	0x7fffffffde5c	0x00007fff
sz	0x7fffffffde58	0xffffde34
n	0x7fffffffde54	0x00000003
	0x7fffffffde50	0x00007fff
arr	0x7fffffffde4c	0xffffde38
	0x7fffffffde48	
	0x7fffffffde44	
	0x7fffffffde40	
	0x7fffffffde3c	
	0x7fffffffde38	
	0x7fffffffde34	0x00000003

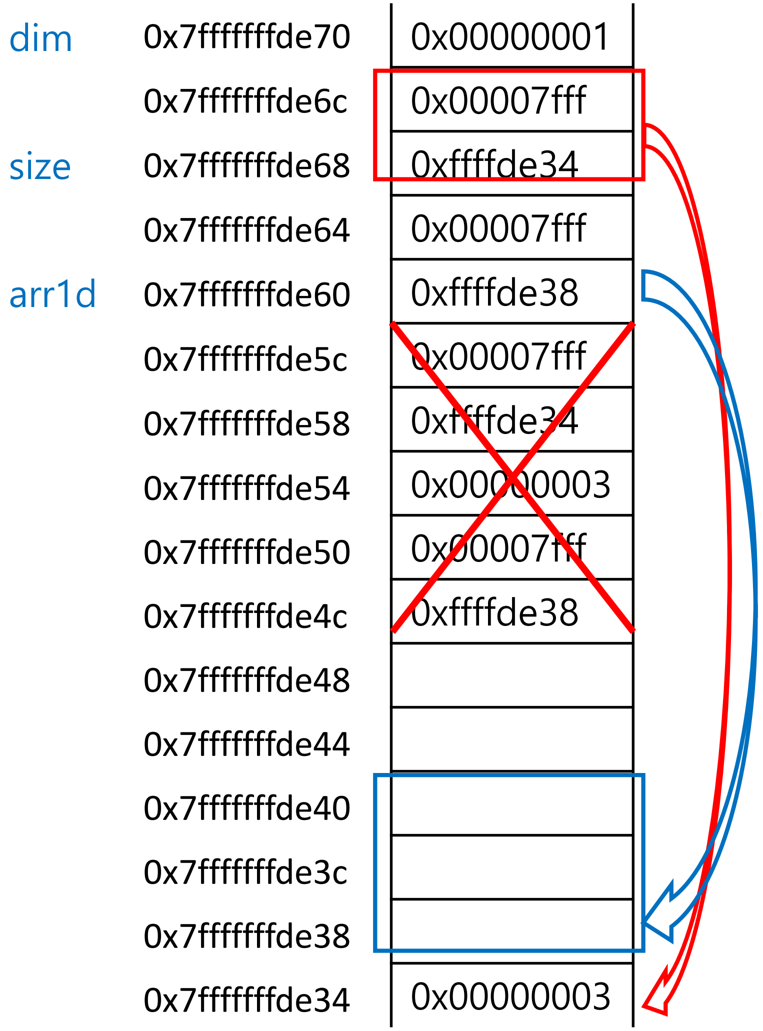
1차원 배열 만들어 반환하기 makeArray1D()

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

int *makeArray1D(int *sz);
void destroyArray1D(int *arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int *arr1d = NULL;

  arr1d = makeArray1D(size);
  for (int i=0; i<size[0]; i++) arr1d[i] = i;
  for (int i=0; i<size[0]; i++) cout << arr1d[i] << " " ;
  cout << endl;
  destroyArray1D(arr1d, size);
  return 0;
}
```

```cpp
int *makeArray1D(int *sz){
  int n = sz[0];
  int *arr = new int[n];
  return arr;
}
void destroyArray1D(int *arr,int *sz){
  delete[] arr;
}
```

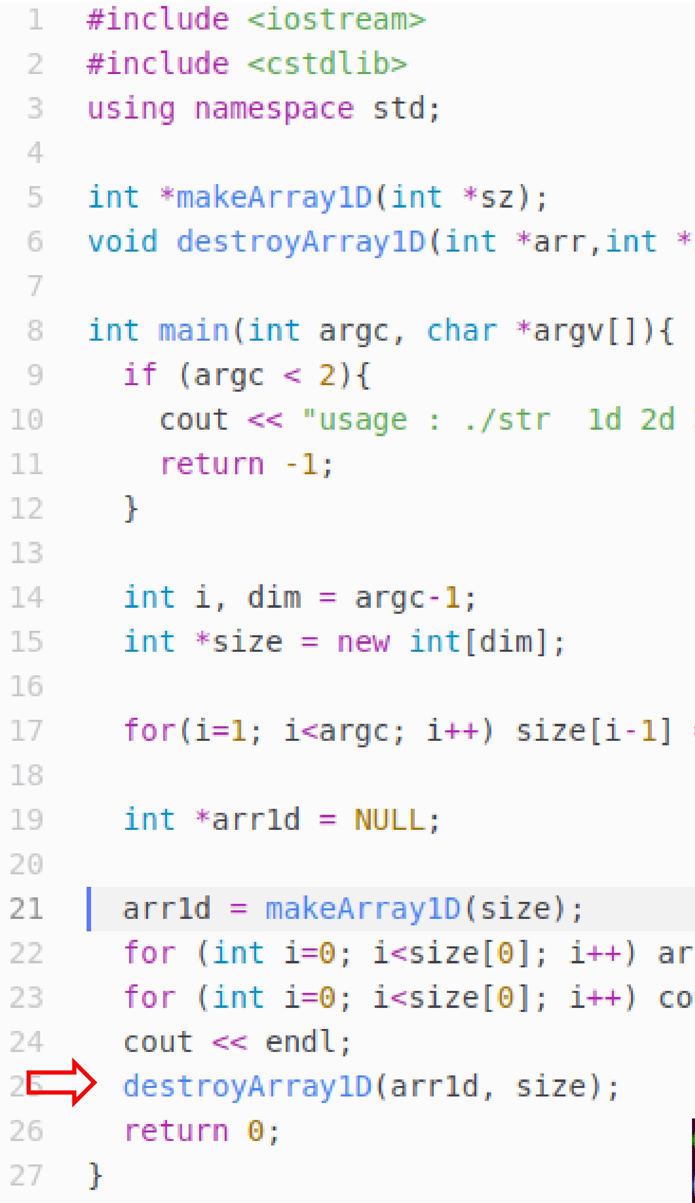


```
ejim@ejim-VirtualBox:~/C2020$ ./alloc1d 3
0 1 2
```

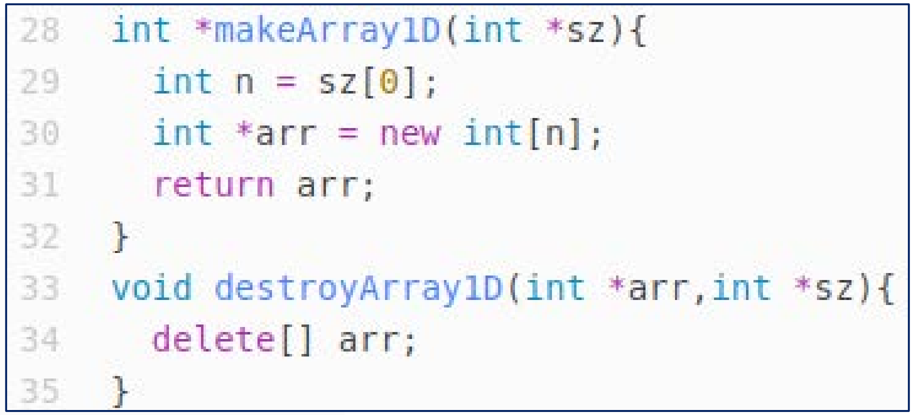


dim	0x7fffffffde70	0x00000001
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde34
	0x7fffffffde64	0x00007fff
arr1d	0x7fffffffde60	0xffffde38
	0x7fffffffde5c	~~0x00007fff~~
	0x7fffffffde58	~~0xffffde34~~
	0x7fffffffde54	~~0x00000003~~
	0x7fffffffde50	~~0x00007fff~~
	0x7fffffffde4c	~~0xffffde38~~
	0x7fffffffde48	
	0x7fffffffde44	
	0x7fffffffde40	
	0x7fffffffde3c	
	0x7fffffffde38	
	0x7fffffffde34	0x00000003

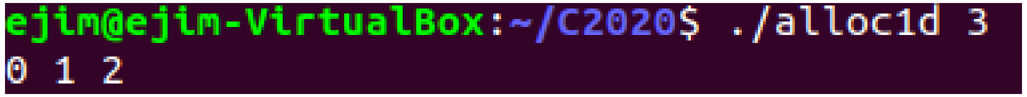
1차원 배열 만들어 반환하기 makeArray1D()



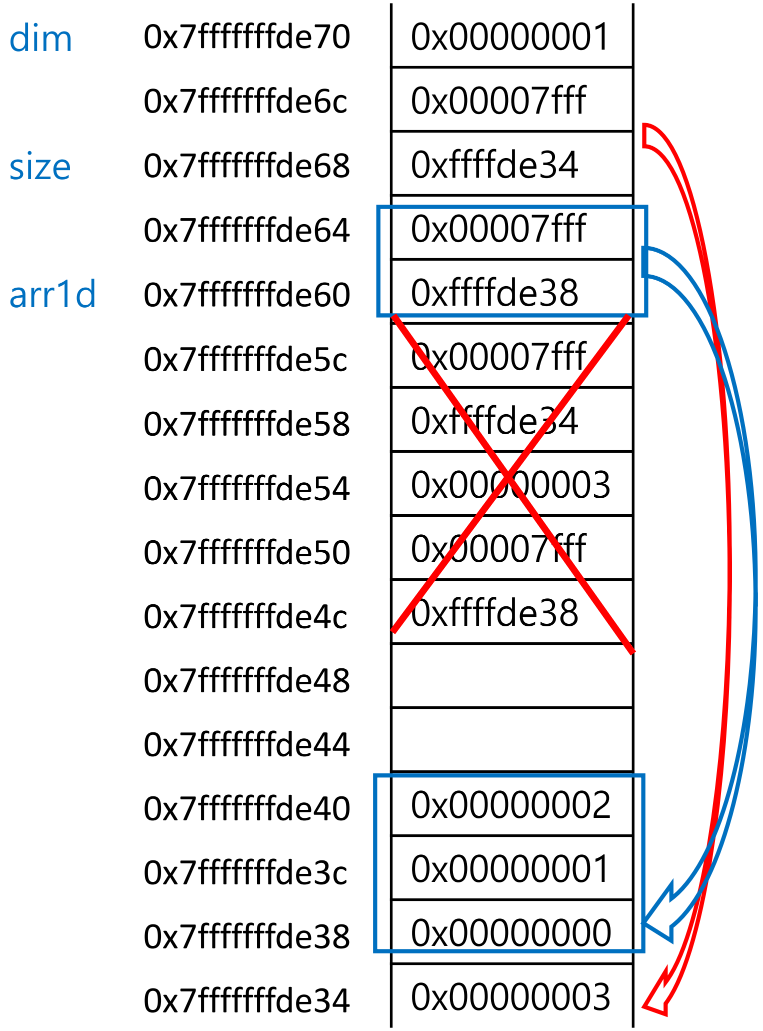
```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int *makeArray1D(int *sz);
void destroyArray1D(int *arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int *arr1d = NULL;

  arr1d = makeArray1D(size);
  for (int i=0; i<size[0]; i++) arr1d[i] = i;
  for (int i=0; i<size[0]; i++) cout << arr1d[i] << " " ;
  cout << endl;
  destroyArray1D(arr1d, size);
  return 0;
}
```



```cpp
int *makeArray1D(int *sz){
  int n = sz[0];
  int *arr = new int[n];
  return arr;
}
void destroyArray1D(int *arr,int *sz){
  delete[] arr;
}
```

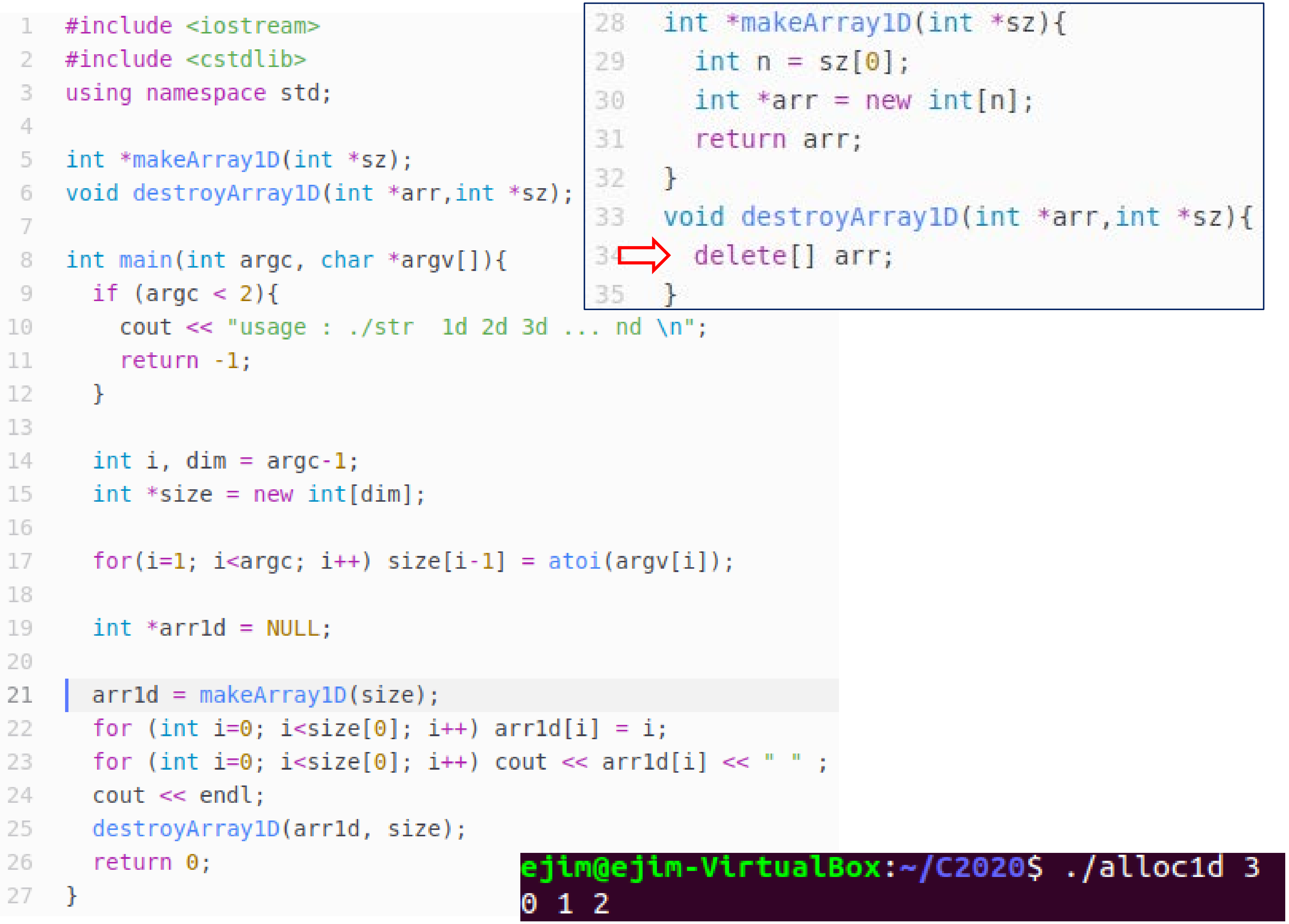


```
ejim@ejim-VirtualBox:~/C2020$ ./alloc1d 3
0 1 2
```



dim	0x7fffffffde70	0x00000001
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde34
	0x7fffffffde64	0x00007fff
arr1d	0x7fffffffde60	0xffffde38
	0x7fffffffde5c	0x00007fff
	0x7fffffffde58	0xffffde34
	0x7fffffffde54	0x00000003
	0x7fffffffde50	0x00007fff
	0x7fffffffde4c	0xffffde38
	0x7fffffffde48	
	0x7fffffffde44	
	0x7fffffffde40	0x00000002
	0x7fffffffde3c	0x00000001
	0x7fffffffde38	0x00000000
	0x7fffffffde34	0x00000003

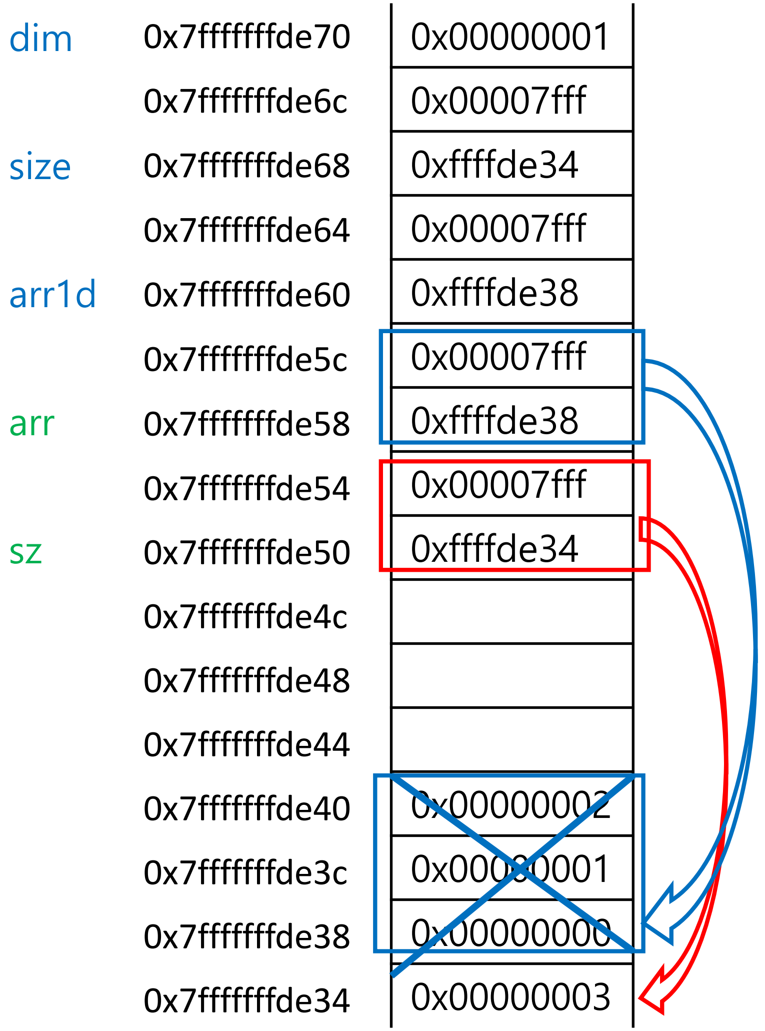
1차원 배열 destroyArray1D()



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

int *makeArray1D(int *sz);
void destroyArray1D(int *arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int *arr1d = NULL;

  arr1d = makeArray1D(size);
  for (int i=0; i<size[0]; i++) arr1d[i] = i;
  for (int i=0; i<size[0]; i++) cout << arr1d[i] << " " ;
  cout << endl;
  destroyArray1D(arr1d, size);
  return 0;
}
```

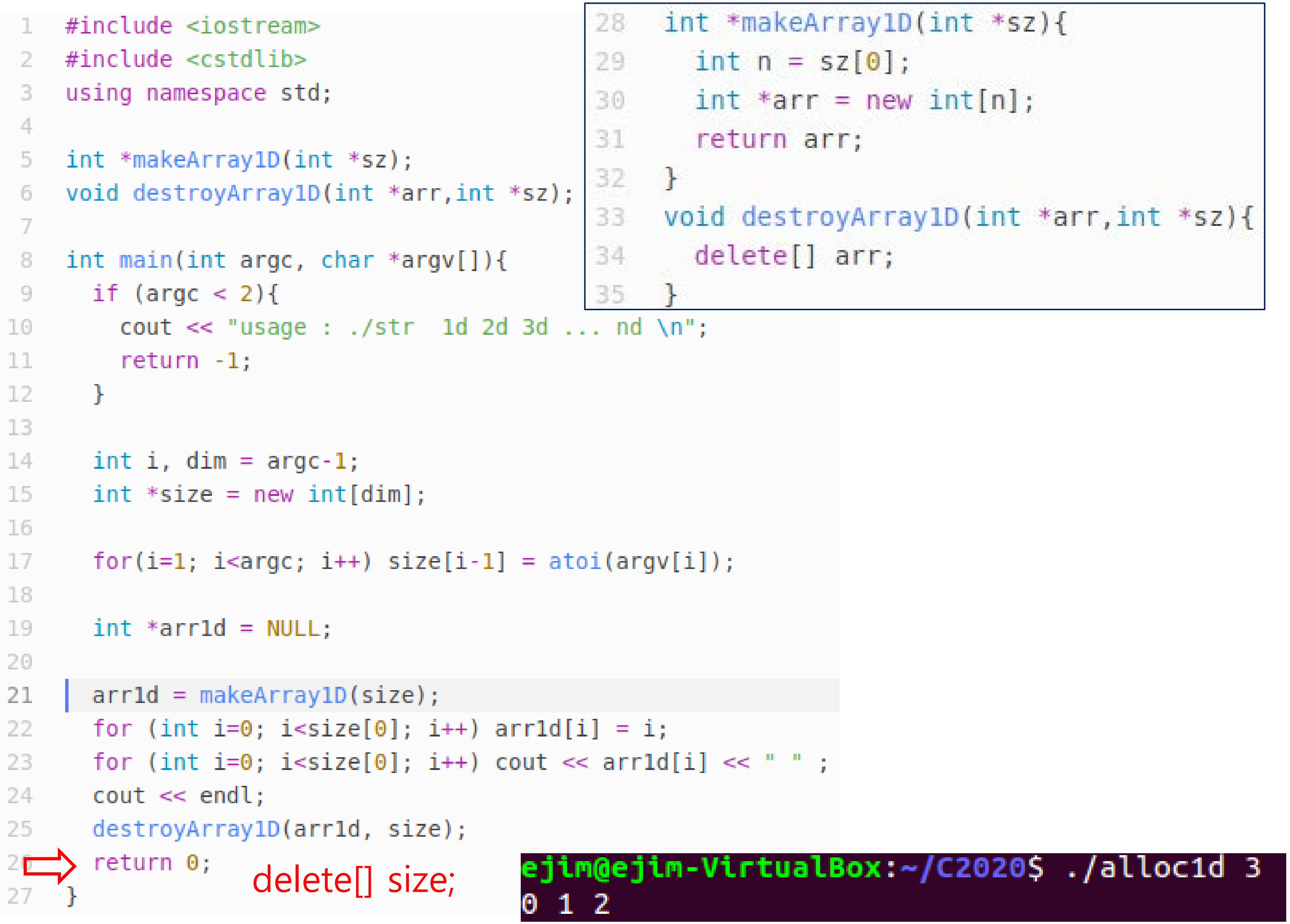
```cpp
int *makeArray1D(int *sz){
  int n = sz[0];
  int *arr = new int[n];
  return arr;
}
void destroyArray1D(int *arr,int *sz){
  delete[] arr;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./alloc1d 3
0 1 2
```



dim	0x7fffffffde70	0x00000001
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde34
	0x7fffffffde64	0x00007fff
arr1d	0x7fffffffde60	0xffffde38
	0x7fffffffde5c	0x00007fff
arr	0x7fffffffde58	0xffffde38
	0x7fffffffde54	0x00007fff
sz	0x7fffffffde50	0xffffde34
	0x7fffffffde4c	
	0x7fffffffde48	
	0x7fffffffde44	
	0x7fffffffde40	~~0x00000002~~
	0x7fffffffde3c	~~0x00000001~~
	0x7fffffffde38	~~0x00000000~~
	0x7fffffffde34	0x00000003

1차원 배열 destroyArray1D()

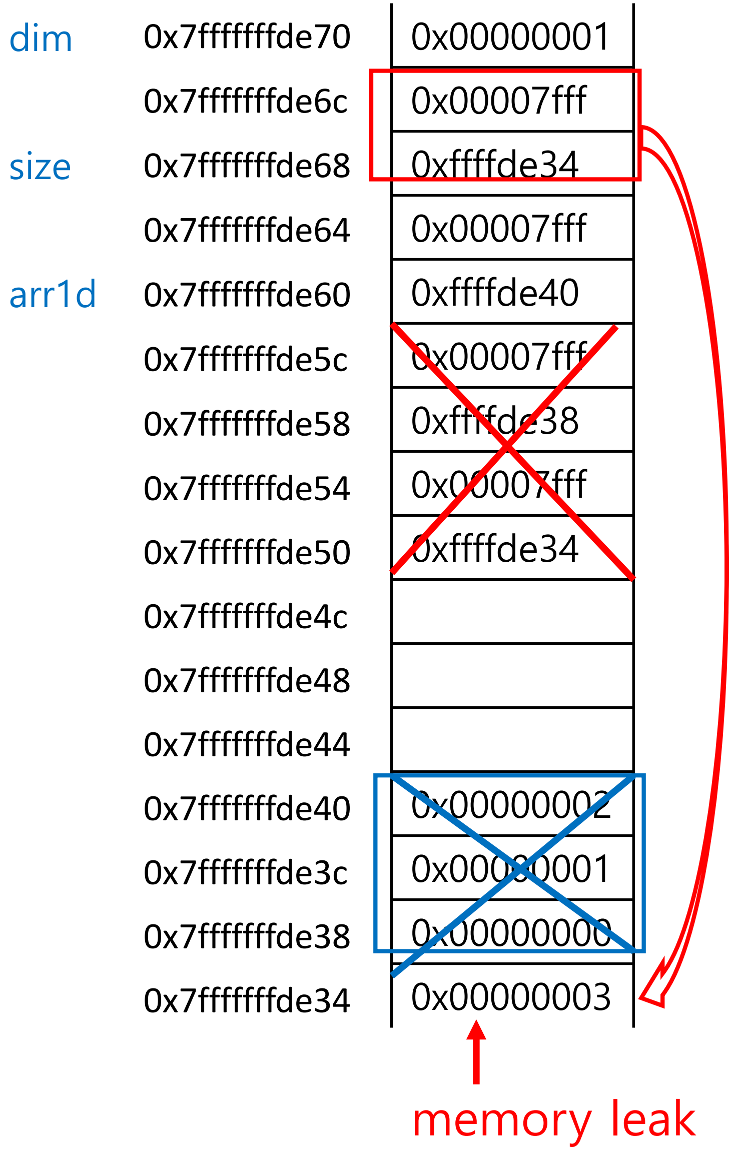


```
#include <iostream>
#include <cstdlib>
using namespace std;

int *makeArray1D(int *sz);
void destroyArray1D(int *arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int *arr1d = NULL;

  arr1d = makeArray1D(size);
  for (int i=0; i<size[0]; i++) arr1d[i] = i;
  for (int i=0; i<size[0]; i++) cout << arr1d[i] << " " ;
  cout << endl;
  destroyArray1D(arr1d, size);
  return 0;
}
```

```
int *makeArray1D(int *sz){
  int n = sz[0];
  int *arr = new int[n];
  return arr;
}
void destroyArray1D(int *arr,int *sz){
  delete[] arr;
}
```

delete[] size;

```
ejim@ejim-VirtualBox:~/C2020$ ./alloc1d 3
0 1 2
```

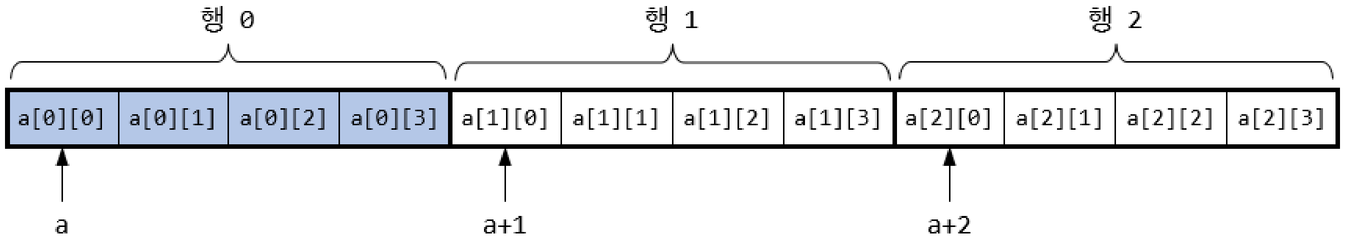


dim	0x7fffffffde70	0x00000001
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde34
	0x7fffffffde64	0x00007fff
arr1d	0x7fffffffde60	0xffffde40
	0x7fffffffde5c	~~0x00007fff~~
	0x7fffffffde58	~~0xffffde38~~
	0x7fffffffde54	~~0x00007fff~~
	0x7fffffffde50	~~0xffffde34~~
	0x7fffffffde4c	
	0x7fffffffde48	
	0x7fffffffde44	
	0x7fffffffde40	~~0x00000002~~
	0x7fffffffde3c	~~0x00000001~~
	0x7fffffffde38	~~0x00000000~~
	0x7fffffffde34	0x00000003

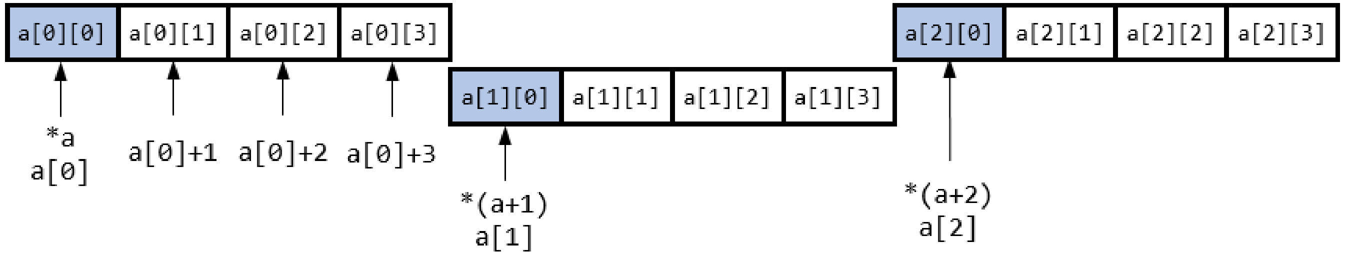
memory leak

a

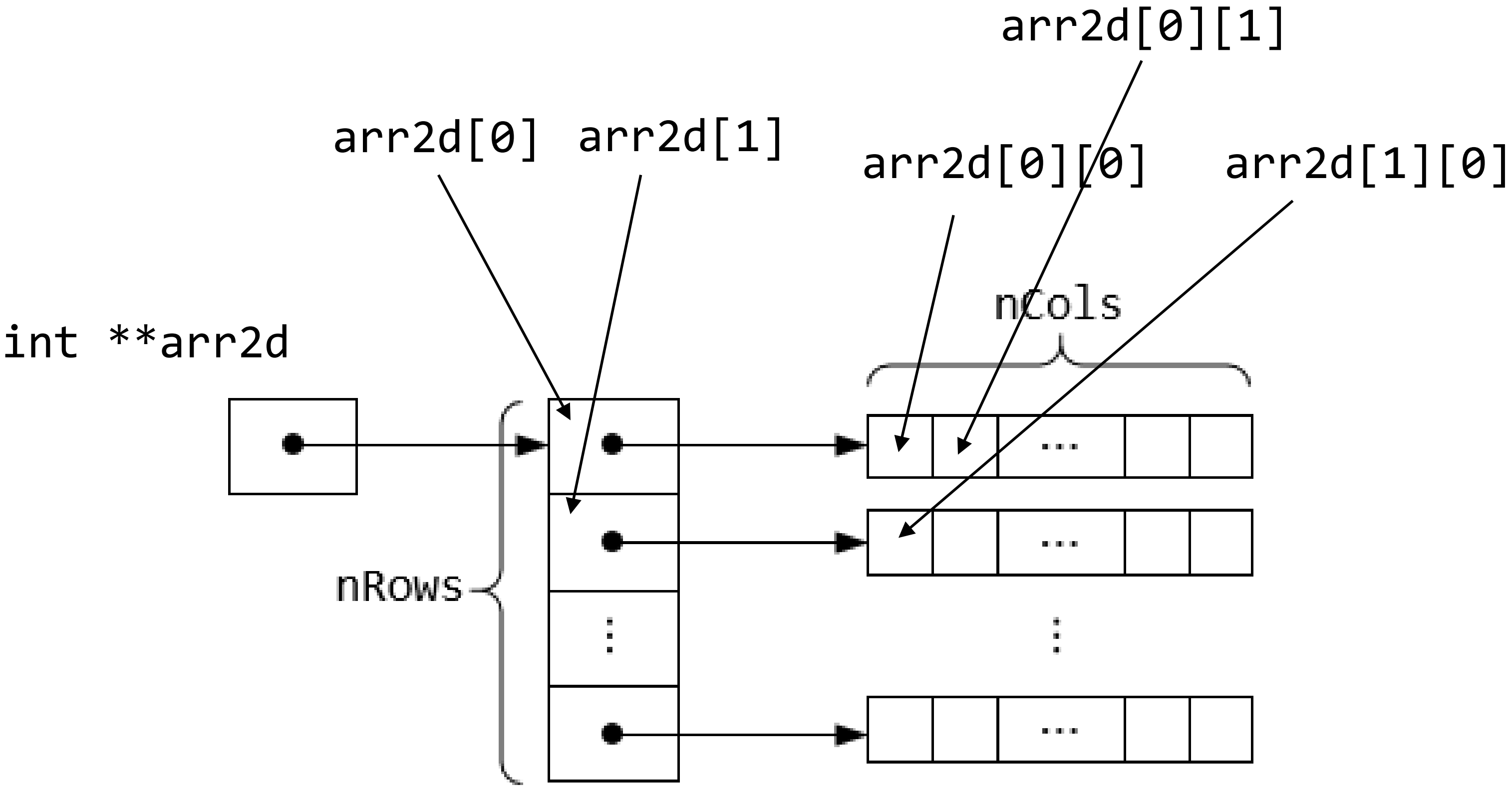
a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]



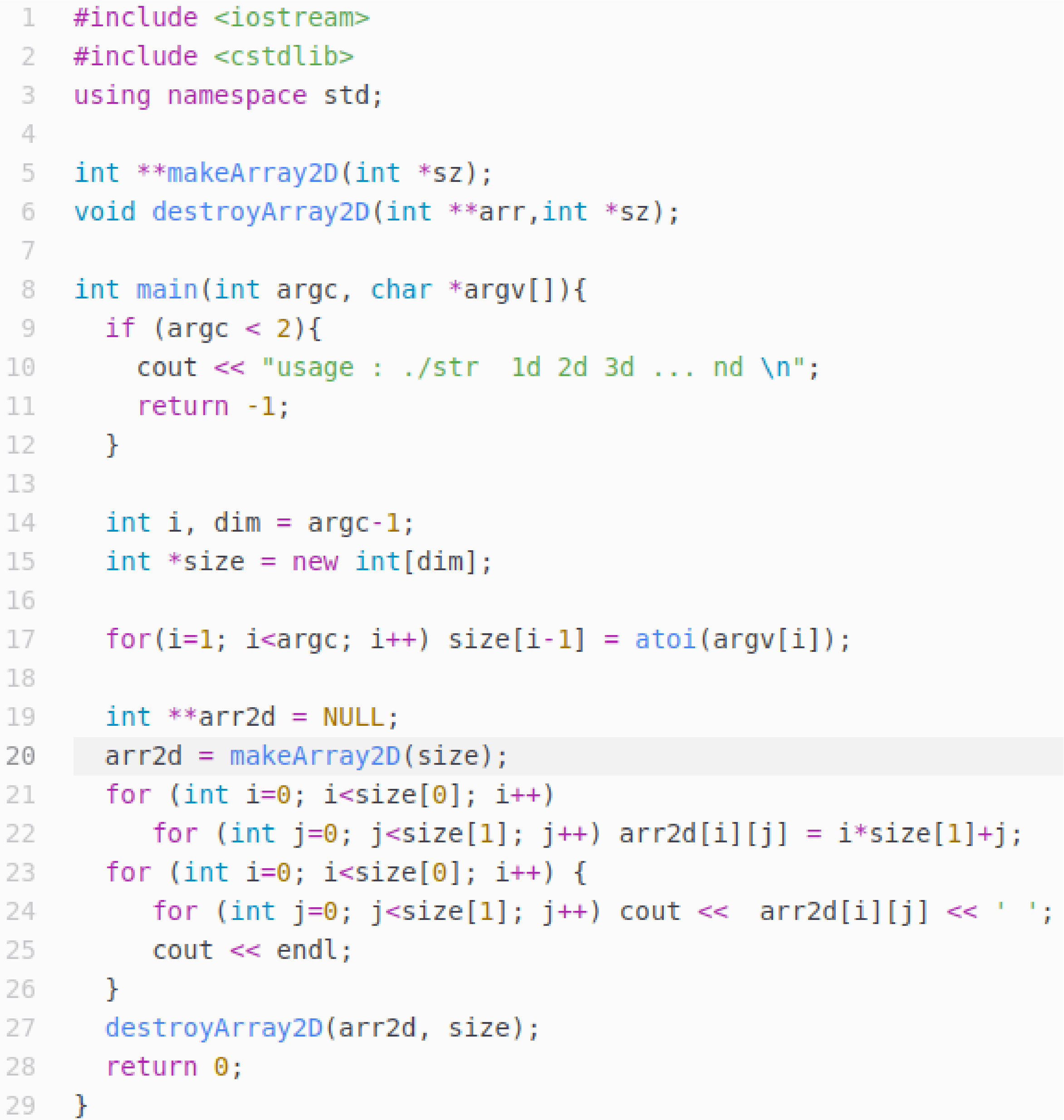
a   : 행을 원소로 하는 1차원 배열의 첫 번째 원소
      즉 첫 번째 행(원소가 4개인 1차원 배열)의 주소
a+1 : 두 번째 행의 주소
a+2 : 세 번째 행의 주소



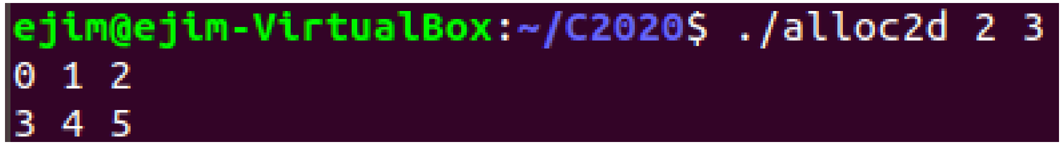
a[0], *a   : 첫 번째 행(원소가 4개인 1차원 배열)의 첫 번째 원소 a[0][0] 의 주소
               (따라서 첫 번째 행인 1차원 배열의 이름으로 사용할 수 있음)
a[1], *(a+1) : 두 번째 행의 첫 번째 원소 a[1][0]의 주소 (두 번째 행인 1차원 배열의 이름)
a[2], *(a+2) : 세 번째 행의 첫 번째 원소 a[2][0]의 주소 (세 번째 행인 1차원 배열의 이름)


arr2d[0][1]
arr2d[0]
arr2d[1]
arr2d[0][0]
arr2d[1][0]
nCols
int **arr2d
nRows
⋮
⋯

2차원 배열 만들어 반환하기 makeArray2D()

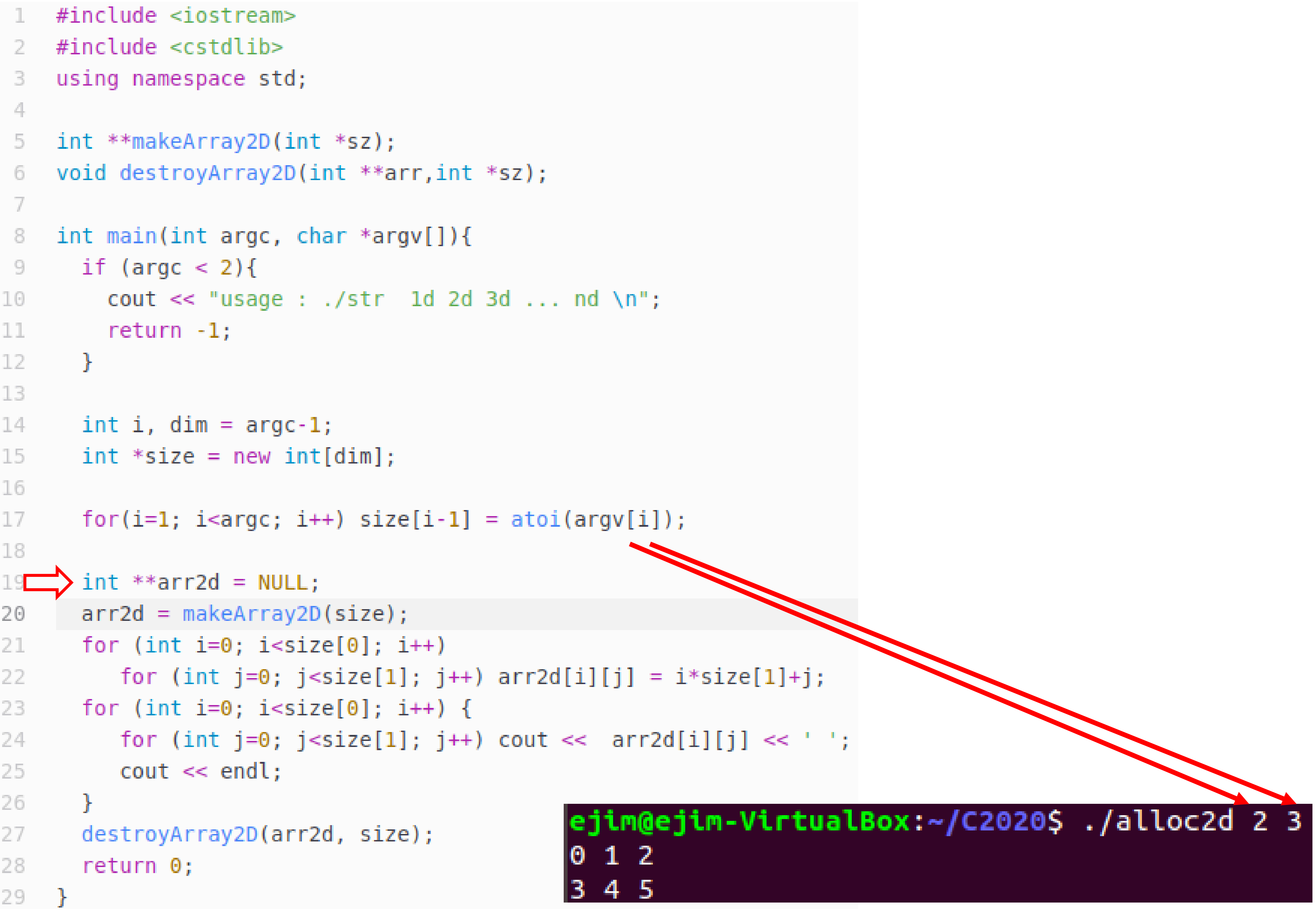


```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int **makeArray2D(int *sz);
void destroyArray2D(int **arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int **arr2d = NULL;
  arr2d = makeArray2D(size);
  for (int i=0; i<size[0]; i++)
     for (int j=0; j<size[1]; j++) arr2d[i][j] = i*size[1]+j;
  for (int i=0; i<size[0]; i++) {
     for (int j=0; j<size[1]; j++) cout <<  arr2d[i][j] << ' ';
     cout << endl;
  }
  destroyArray2D(arr2d, size);
  return 0;
}
```



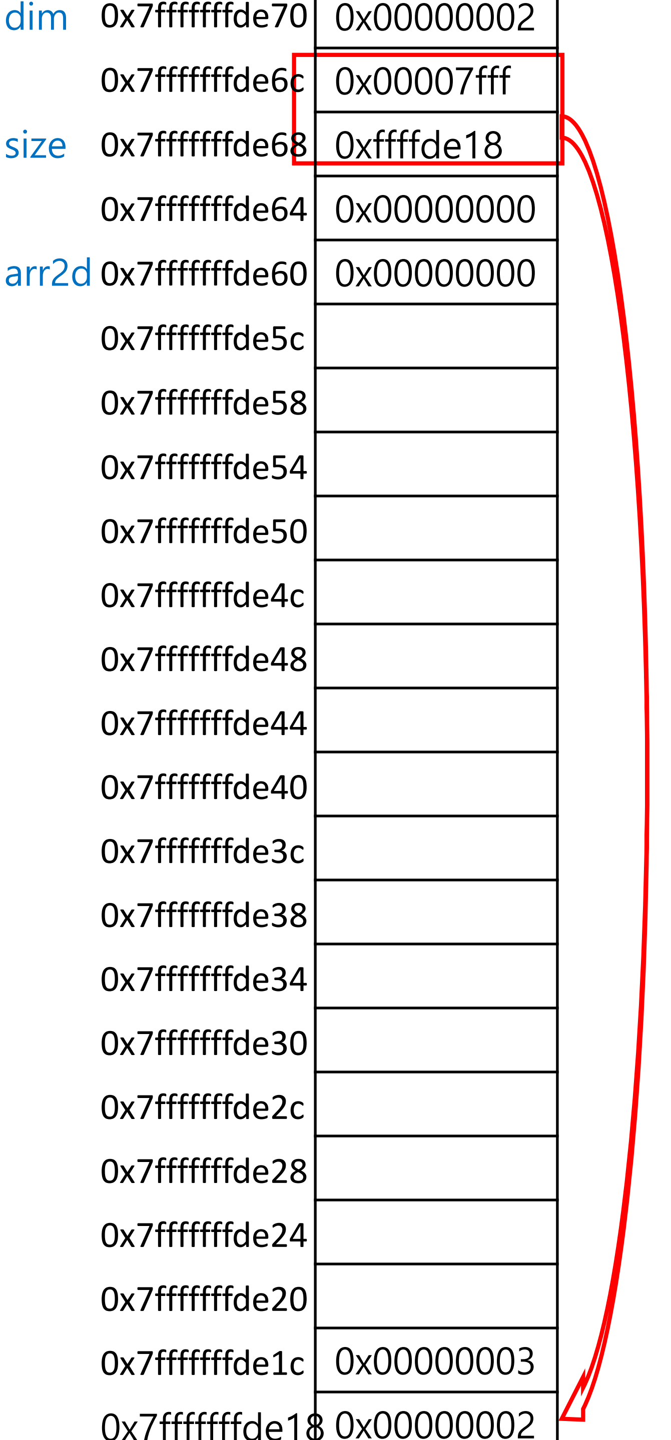
```
ejim@ejim-VirtualBox:~/C2020$ ./alloc2d 2 3
0 1 2
3 4 5
```

2차원 배열 만들어 반환하기 makeArray2D()



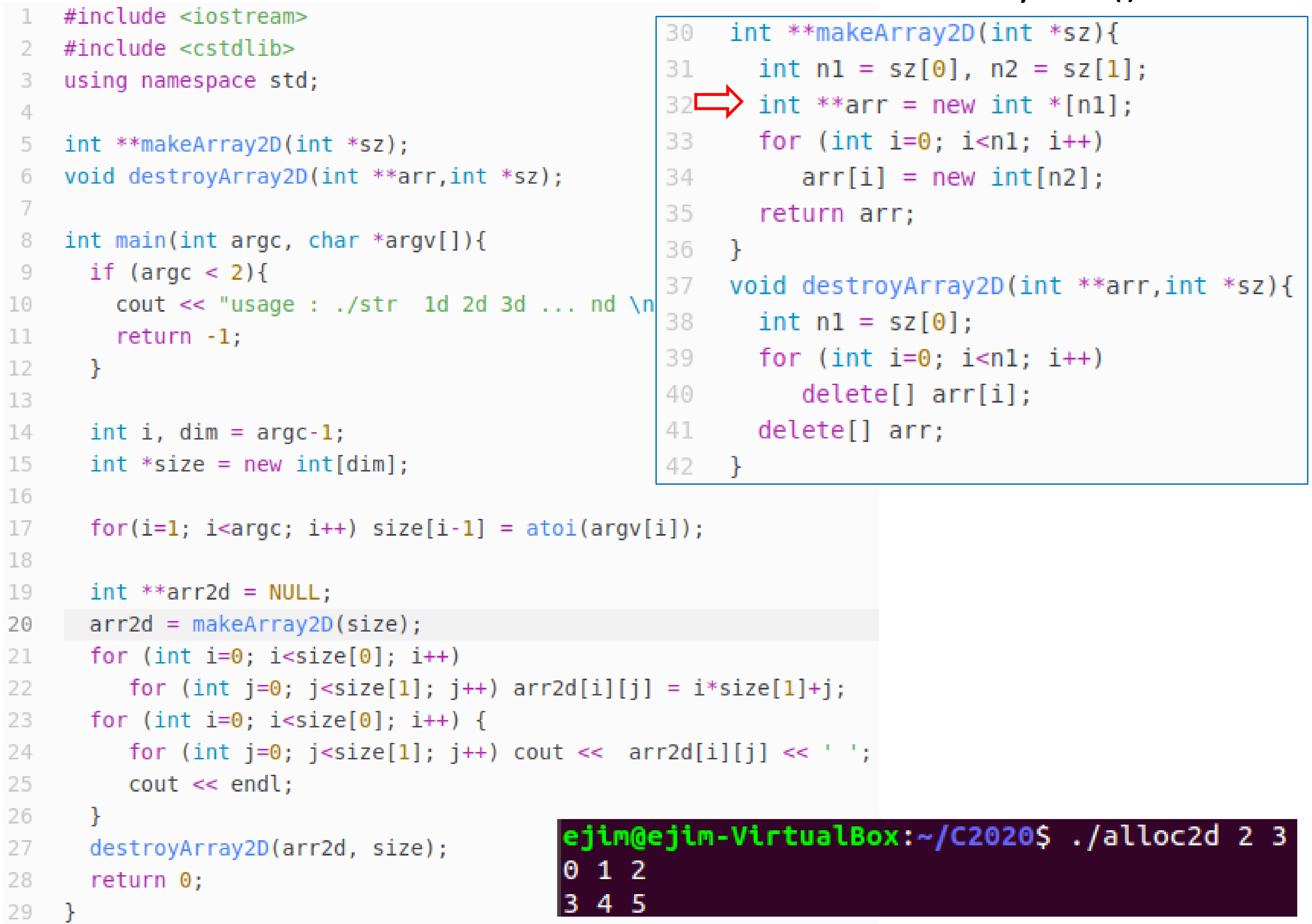
```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int **makeArray2D(int *sz);
void destroyArray2D(int **arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int **arr2d = NULL;
  arr2d = makeArray2D(size);
  for (int i=0; i<size[0]; i++)
     for (int j=0; j<size[1]; j++) arr2d[i][j] = i*size[1]+j;
  for (int i=0; i<size[0]; i++) {
     for (int j=0; j<size[1]; j++) cout <<  arr2d[i][j] << ' ';
     cout << endl;
  }
  destroyArray2D(arr2d, size);
  return 0;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./alloc2d 2 3
0 1 2
3 4 5
```



dim	0x7fffffffde70	0x00000002
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde18
	0x7fffffffde64	0x00000000
arr2d	0x7fffffffde60	0x00000000
	0x7fffffffde5c	
	0x7fffffffde58	
	0x7fffffffde54	
	0x7fffffffde50	
	0x7fffffffde4c	
	0x7fffffffde48	
	0x7fffffffde44	
	0x7fffffffde40	
	0x7fffffffde3c	
	0x7fffffffde38	
	0x7fffffffde34	
	0x7fffffffde30	
	0x7fffffffde2c	
	0x7fffffffde28	
	0x7fffffffde24	
	0x7fffffffde20	
	0x7fffffffde1c	0x00000003
	0x7fffffffde18	0x00000002

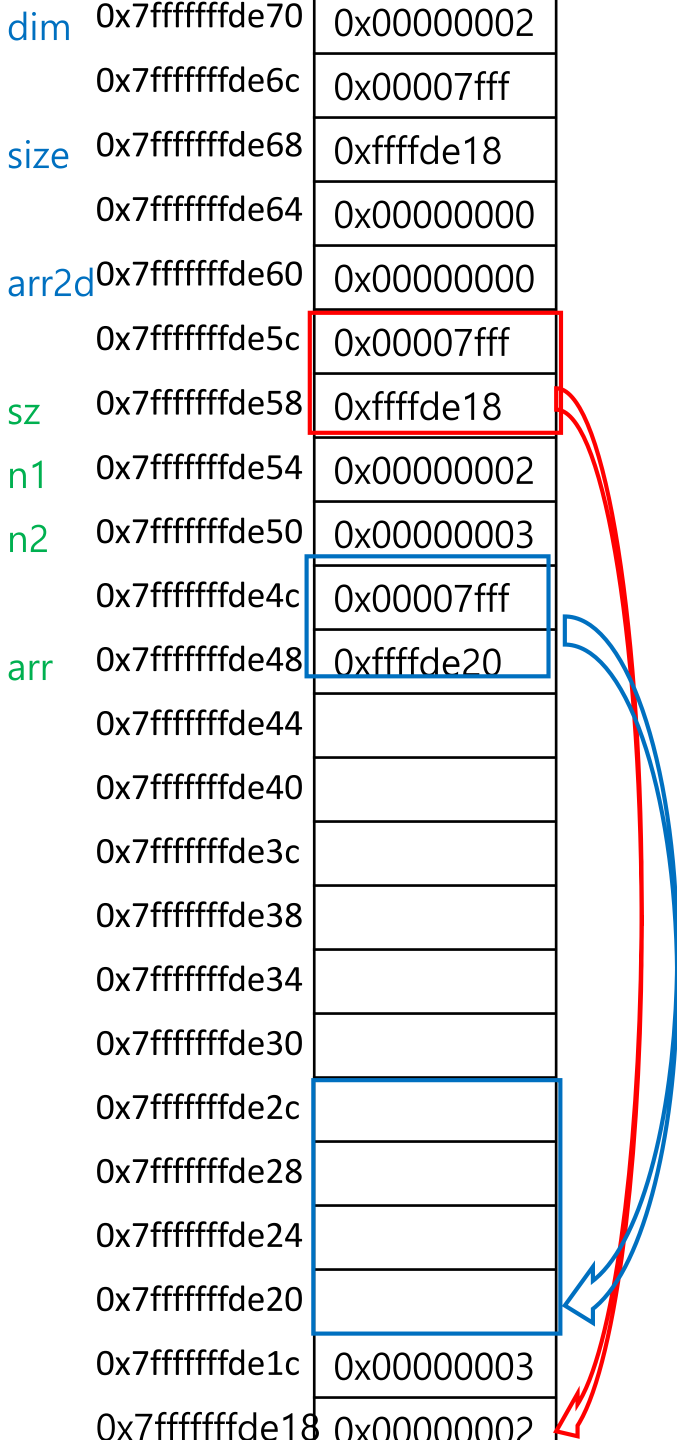
2차원 배열 만들어 반환하기 makeArray2D()



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

int **makeArray2D(int *sz);
void destroyArray2D(int **arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int **arr2d = NULL;
  arr2d = makeArray2D(size);
  for (int i=0; i<size[0]; i++)
     for (int j=0; j<size[1]; j++) arr2d[i][j] = i*size[1]+j;
  for (int i=0; i<size[0]; i++) {
     for (int j=0; j<size[1]; j++) cout <<  arr2d[i][j] << ' ';
     cout << endl;
  }
  destroyArray2D(arr2d, size);
  return 0;
}
```

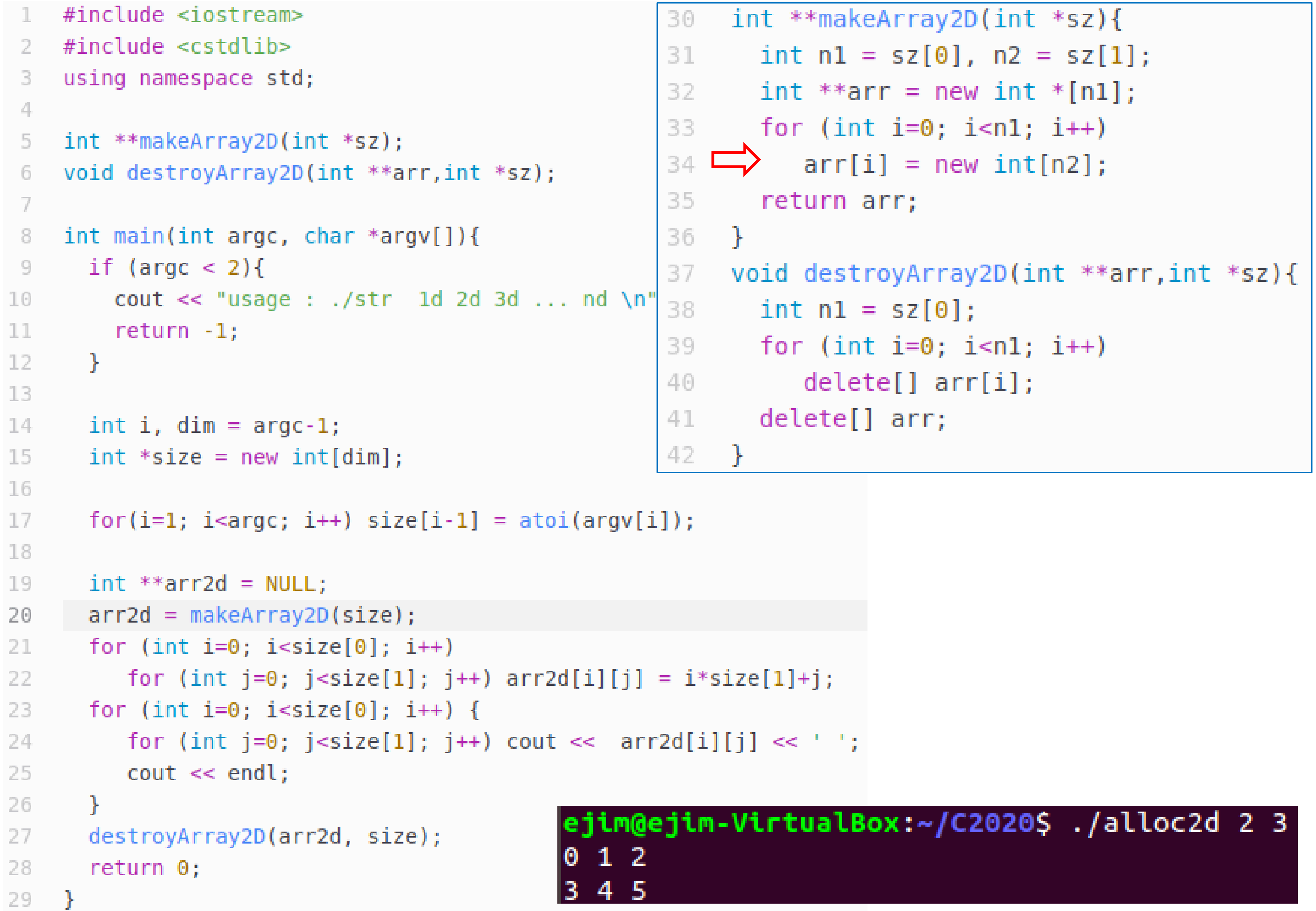
```cpp
int **makeArray2D(int *sz){
  int n1 = sz[0], n2 = sz[1];
⇨ int **arr = new int *[n1];
  for (int i=0; i<n1; i++)
     arr[i] = new int[n2];
  return arr;
}
void destroyArray2D(int **arr,int *sz){
  int n1 = sz[0];
  for (int i=0; i<n1; i++)
     delete[] arr[i];
  delete[] arr;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./alloc2d 2 3
0 1 2
3 4 5
```



dim	0x7fffffffde70	0x00000002
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde18
	0x7fffffffde64	0x00000000
arr2d	0x7fffffffde60	0x00000000
	0x7fffffffde5c	0x00007fff
sz	0x7fffffffde58	0xffffde18
n1	0x7fffffffde54	0x00000002
n2	0x7fffffffde50	0x00000003
	0x7fffffffde4c	0x00007fff
arr	0x7fffffffde48	0xffffde20
	0x7fffffffde44	
	0x7fffffffde40	
	0x7fffffffde3c	
	0x7fffffffde38	
	0x7fffffffde34	
	0x7fffffffde30	
	0x7fffffffde2c	
	0x7fffffffde28	
	0x7fffffffde24	
	0x7fffffffde20	
	0x7fffffffde1c	0x00000003
	0x7fffffffde18	0x00000002

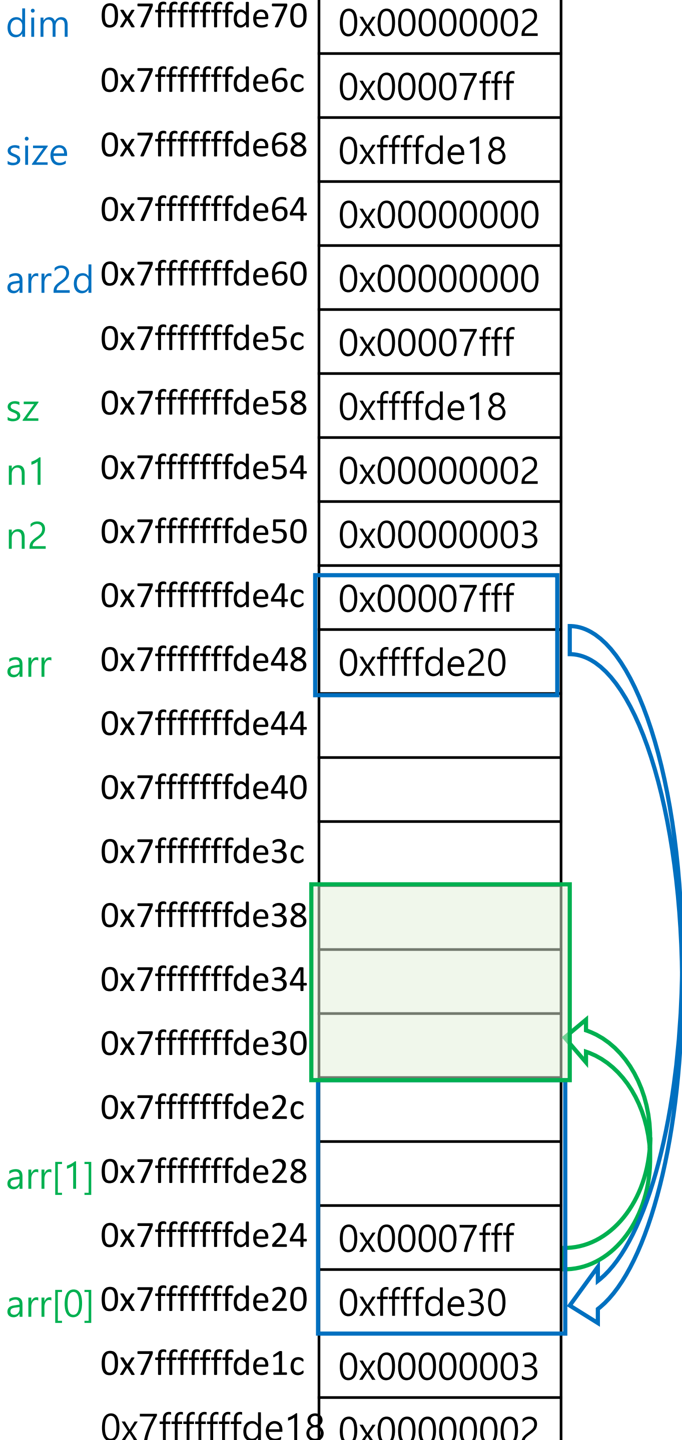
2차원 배열 만들어 반환하기 makeArray2D()



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

int **makeArray2D(int *sz);
void destroyArray2D(int **arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n"
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int **arr2d = NULL;
  arr2d = makeArray2D(size);
  for (int i=0; i<size[0]; i++)
     for (int j=0; j<size[1]; j++) arr2d[i][j] = i*size[1]+j;
  for (int i=0; i<size[0]; i++) {
     for (int j=0; j<size[1]; j++) cout <<  arr2d[i][j] << ' ';
     cout << endl;
  }
  destroyArray2D(arr2d, size);
  return 0;
}
```

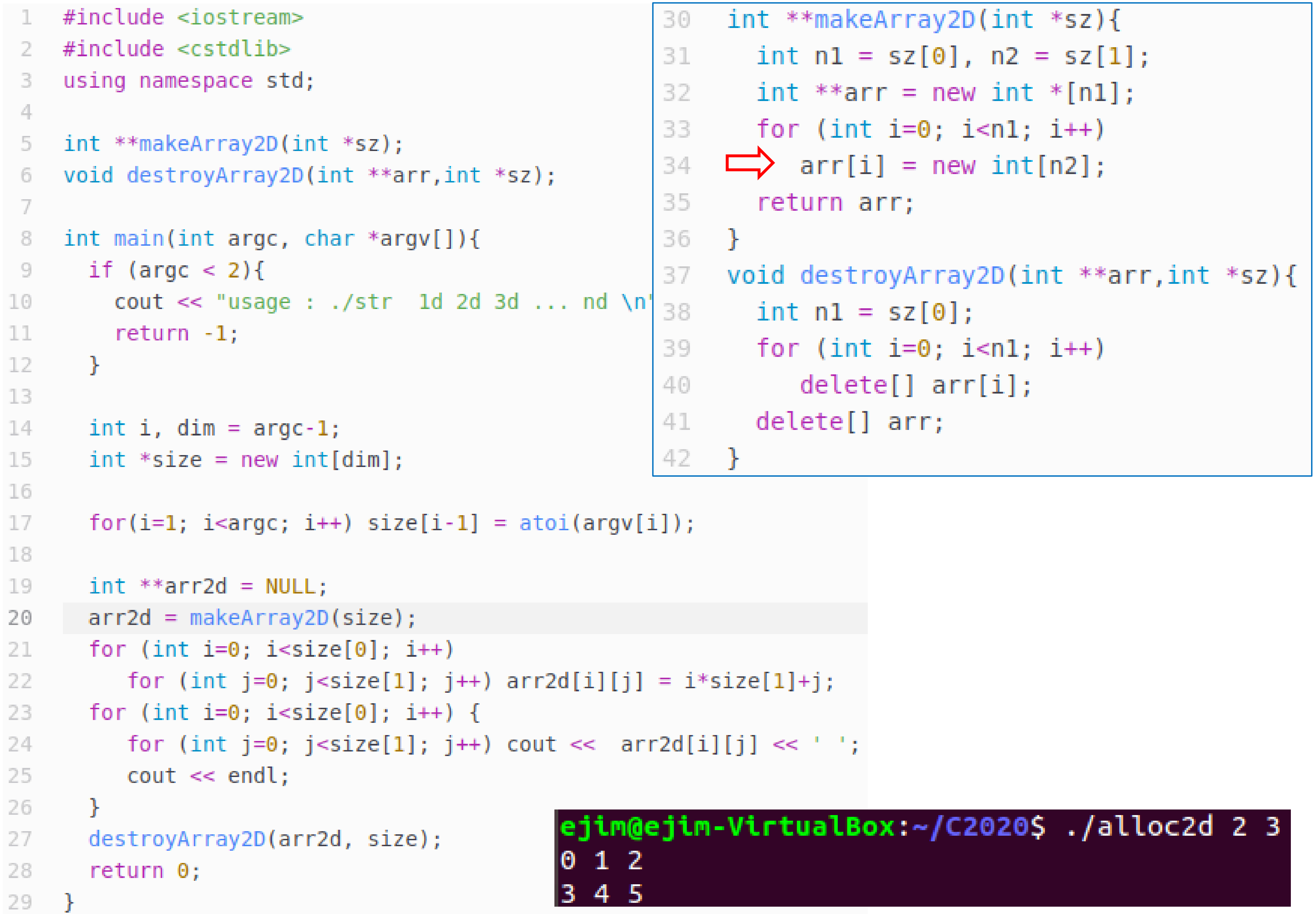
```cpp
int **makeArray2D(int *sz){
  int n1 = sz[0], n2 = sz[1];
  int **arr = new int *[n1];
  for (int i=0; i<n1; i++)
     arr[i] = new int[n2];
  return arr;
}
void destroyArray2D(int **arr,int *sz){
  int n1 = sz[0];
  for (int i=0; i<n1; i++)
     delete[] arr[i];
  delete[] arr;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./alloc2d 2 3
0 1 2
3 4 5
```



dim	0x7fffffffde70	0x00000002
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde18
	0x7fffffffde64	0x00000000
arr2d	0x7fffffffde60	0x00000000
	0x7fffffffde5c	0x00007fff
sz	0x7fffffffde58	0xffffde18
n1	0x7fffffffde54	0x00000002
n2	0x7fffffffde50	0x00000003
	0x7fffffffde4c	0x00007fff
arr	0x7fffffffde48	0xffffde20
	0x7fffffffde44	
	0x7fffffffde40	
	0x7fffffffde3c	
	0x7fffffffde38	
	0x7fffffffde34	
	0x7fffffffde30	
	0x7fffffffde2c	
arr[1]	0x7fffffffde28	
	0x7fffffffde24	0x00007fff
arr[0]	0x7fffffffde20	0xffffde30
	0x7fffffffde1c	0x00000003
	0x7fffffffde18	0x00000002

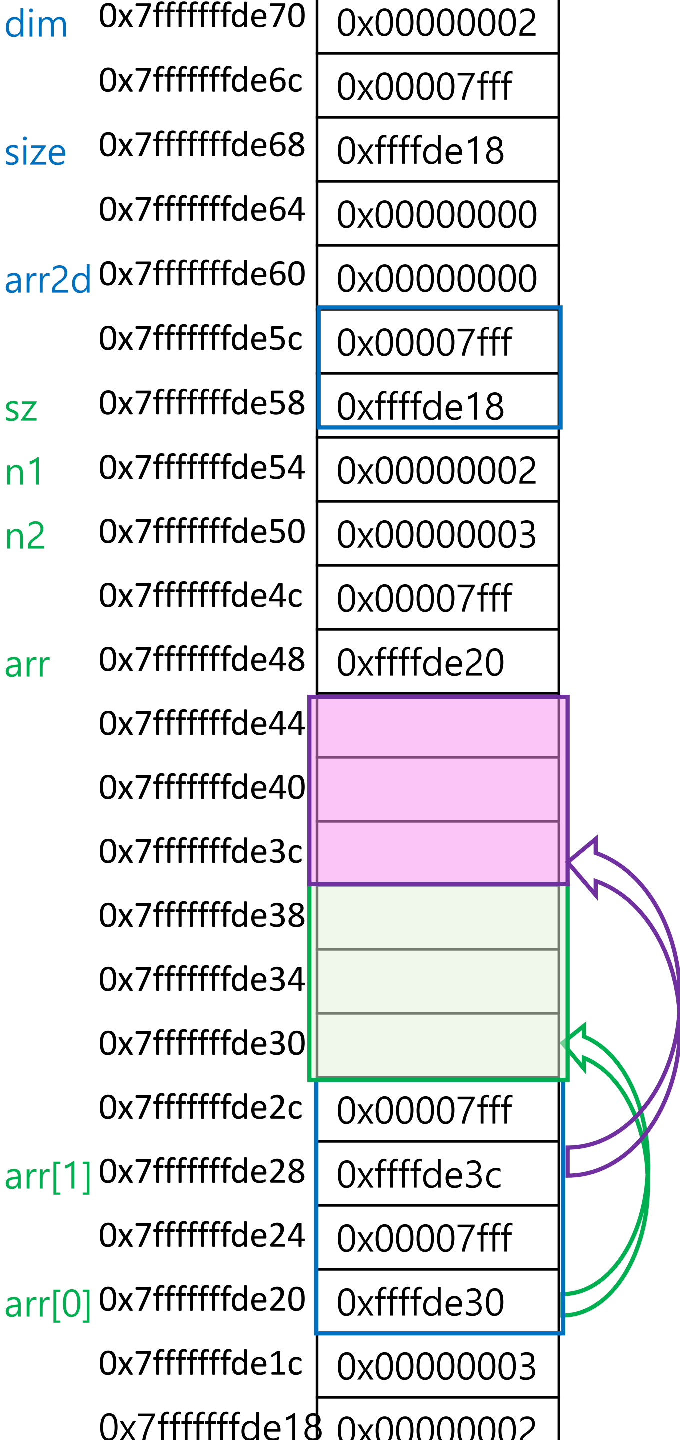
2차원 배열 만들어 반환하기 makeArray2D()



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

int **makeArray2D(int *sz);
void destroyArray2D(int **arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int **arr2d = NULL;
  arr2d = makeArray2D(size);
  for (int i=0; i<size[0]; i++)
     for (int j=0; j<size[1]; j++) arr2d[i][j] = i*size[1]+j;
  for (int i=0; i<size[0]; i++) {
     for (int j=0; j<size[1]; j++) cout <<  arr2d[i][j] << ' ';
     cout << endl;
  }
  destroyArray2D(arr2d, size);
  return 0;
}
```

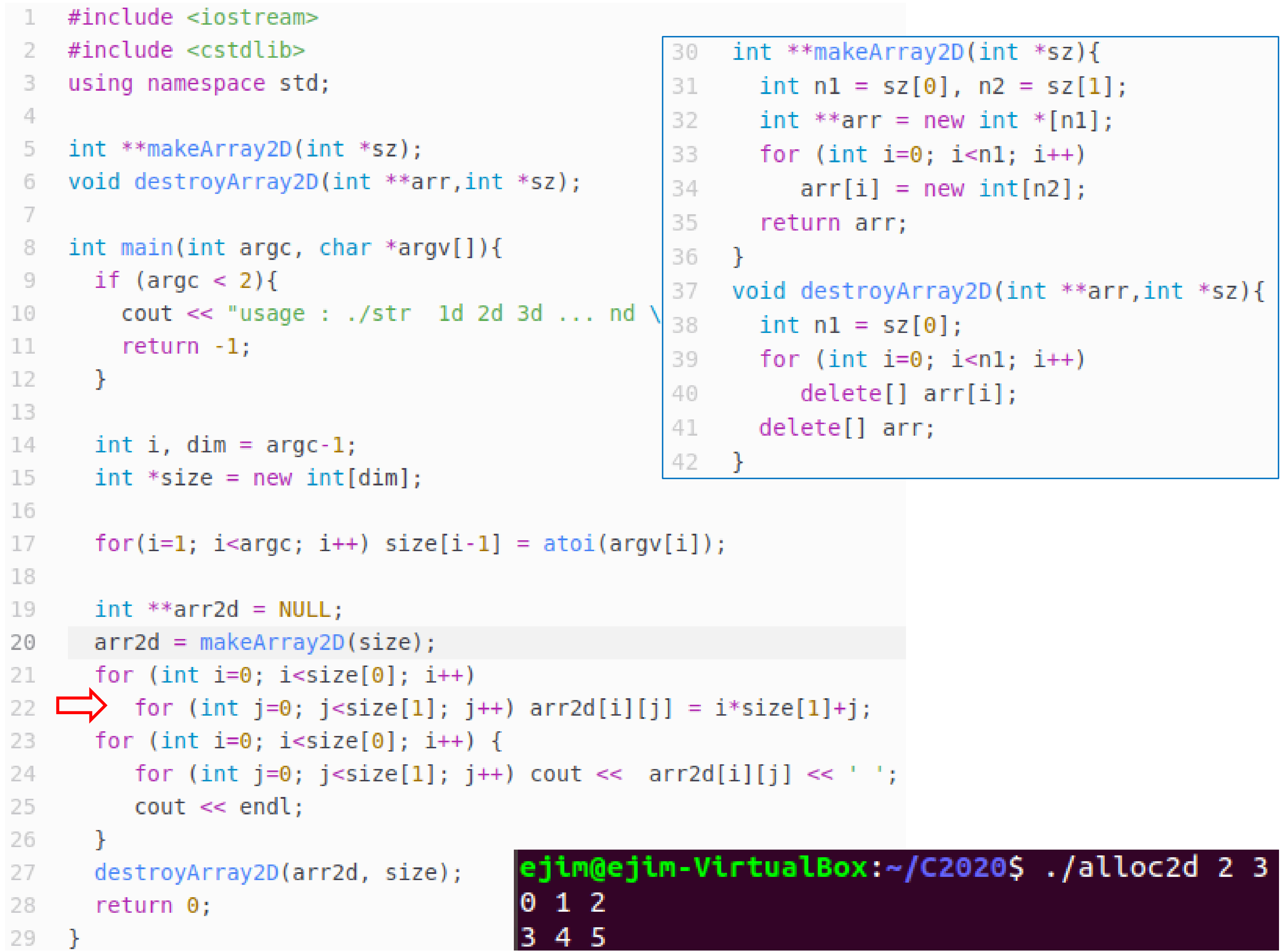
```cpp
int **makeArray2D(int *sz){
  int n1 = sz[0], n2 = sz[1];
  int **arr = new int *[n1];
  for (int i=0; i<n1; i++)
 ⇨  arr[i] = new int[n2];
  return arr;
}
void destroyArray2D(int **arr,int *sz){
  int n1 = sz[0];
  for (int i=0; i<n1; i++)
     delete[] arr[i];
  delete[] arr;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./alloc2d 2 3
0 1 2
3 4 5
```



dim	0x7fffffffde70	0x00000002
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde18
	0x7fffffffde64	0x00000000
arr2d	0x7fffffffde60	0x00000000
	0x7fffffffde5c	0x00007fff
sz	0x7fffffffde58	0xffffde18
n1	0x7fffffffde54	0x00000002
n2	0x7fffffffde50	0x00000003
	0x7fffffffde4c	0x00007fff
arr	0x7fffffffde48	0xffffde20
	0x7fffffffde44	
	0x7fffffffde40	
	0x7fffffffde3c	
	0x7fffffffde38	
	0x7fffffffde34	
	0x7fffffffde30	
	0x7fffffffde2c	0x00007fff
arr[1]	0x7fffffffde28	0xffffde3c
	0x7fffffffde24	0x00007fff
arr[0]	0x7fffffffde20	0xffffde30
	0x7fffffffde1c	0x00000003
	0x7fffffffde18	0x00000002

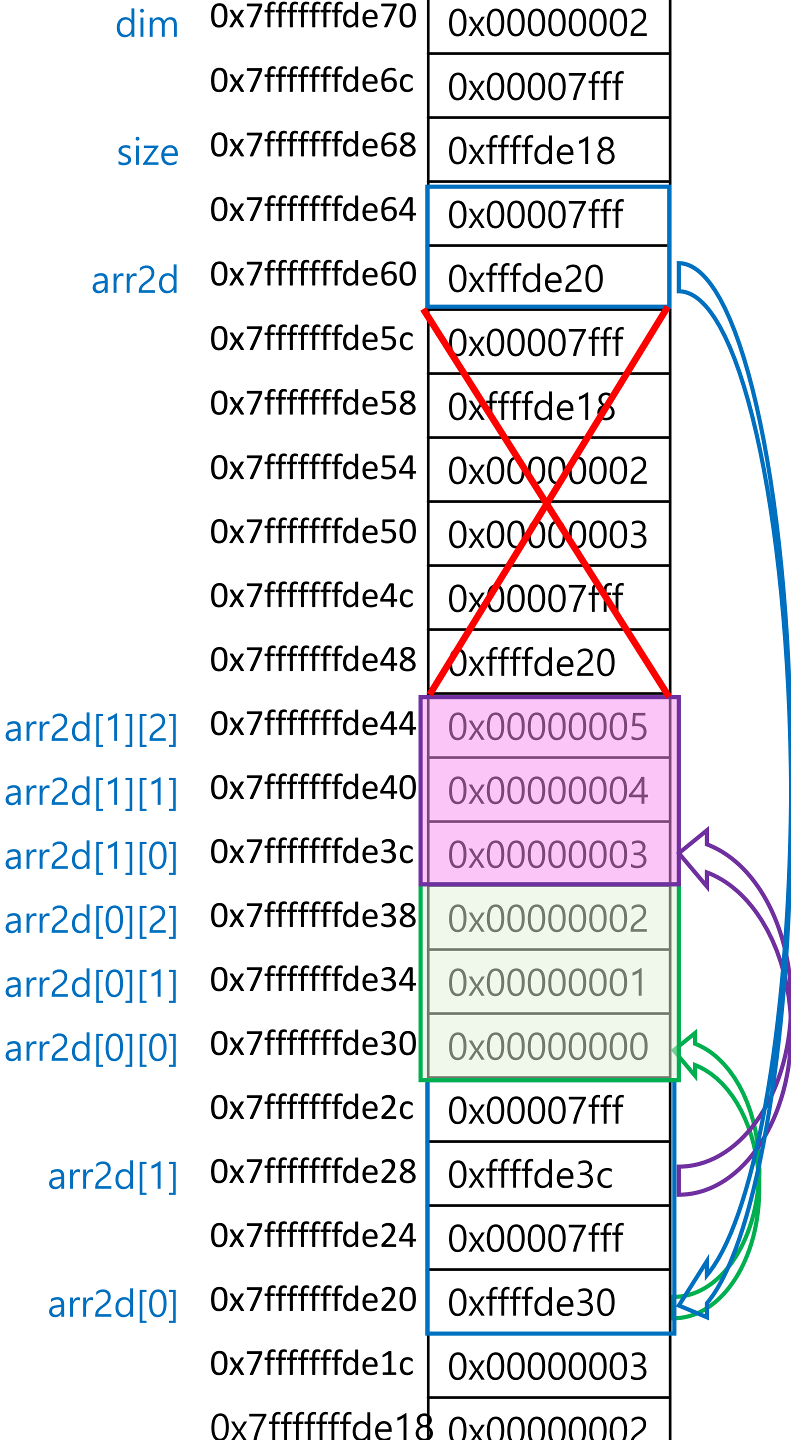
2차원 배열 만들어 반환하기 makeArray2D()



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

int **makeArray2D(int *sz);
void destroyArray2D(int **arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int **arr2d = NULL;
  arr2d = makeArray2D(size);
  for (int i=0; i<size[0]; i++)
    for (int j=0; j<size[1]; j++) arr2d[i][j] = i*size[1]+j;
  for (int i=0; i<size[0]; i++) {
    for (int j=0; j<size[1]; j++) cout <<  arr2d[i][j] << ' ';
    cout << endl;
  }
  destroyArray2D(arr2d, size);
  return 0;
}
```

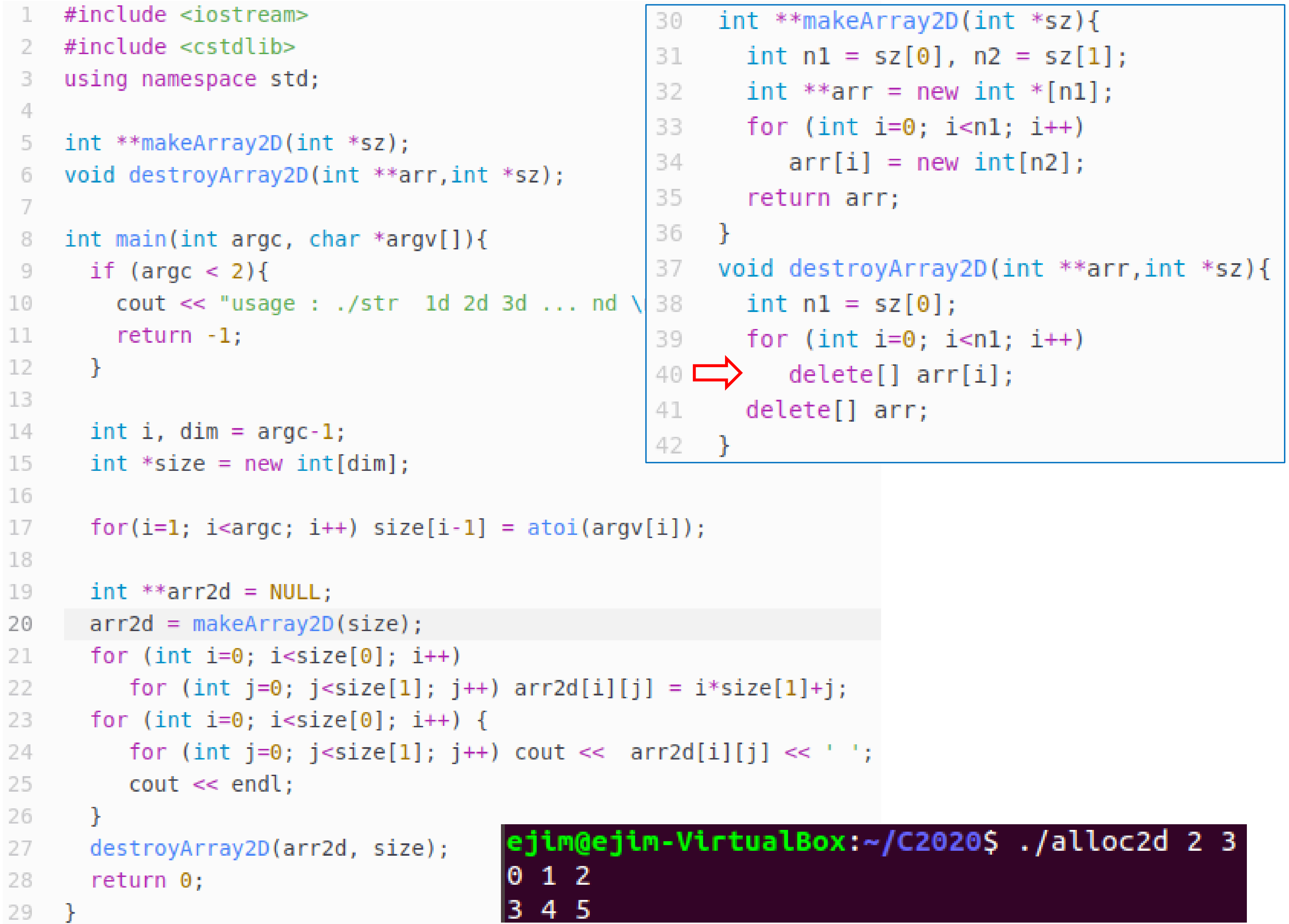
```cpp
int **makeArray2D(int *sz){
  int n1 = sz[0], n2 = sz[1];
  int **arr = new int *[n1];
  for (int i=0; i<n1; i++)
     arr[i] = new int[n2];
  return arr;
}
void destroyArray2D(int **arr,int *sz){
  int n1 = sz[0];
  for (int i=0; i<n1; i++)
     delete[] arr[i];
  delete[] arr;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./alloc2d 2 3
0 1 2
3 4 5
```



dim	0x7fffffffde70	0x00000002
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde18
	0x7fffffffde64	0x00007fff
arr2d	0x7fffffffde60	0xfffde20
	0x7fffffffde5c	~~0x00007fff~~
	0x7fffffffde58	~~0xffffde18~~
	0x7fffffffde54	~~0x00000002~~
	0x7fffffffde50	~~0x00000003~~
	0x7fffffffde4c	~~0x00007fff~~
	0x7fffffffde48	~~0xffffde20~~
arr2d[1][2]	0x7fffffffde44	0x00000005
arr2d[1][1]	0x7fffffffde40	0x00000004
arr2d[1][0]	0x7fffffffde3c	0x00000003
arr2d[0][2]	0x7fffffffde38	0x00000002
arr2d[0][1]	0x7fffffffde34	0x00000001
arr2d[0][0]	0x7fffffffde30	0x00000000
	0x7fffffffde2c	0x00007fff
arr2d[1]	0x7fffffffde28	0xffffde3c
	0x7fffffffde24	0x00007fff
arr2d[0]	0x7fffffffde20	0xffffde30
	0x7fffffffde1c	0x00000003
	0x7fffffffde18	0x00000002

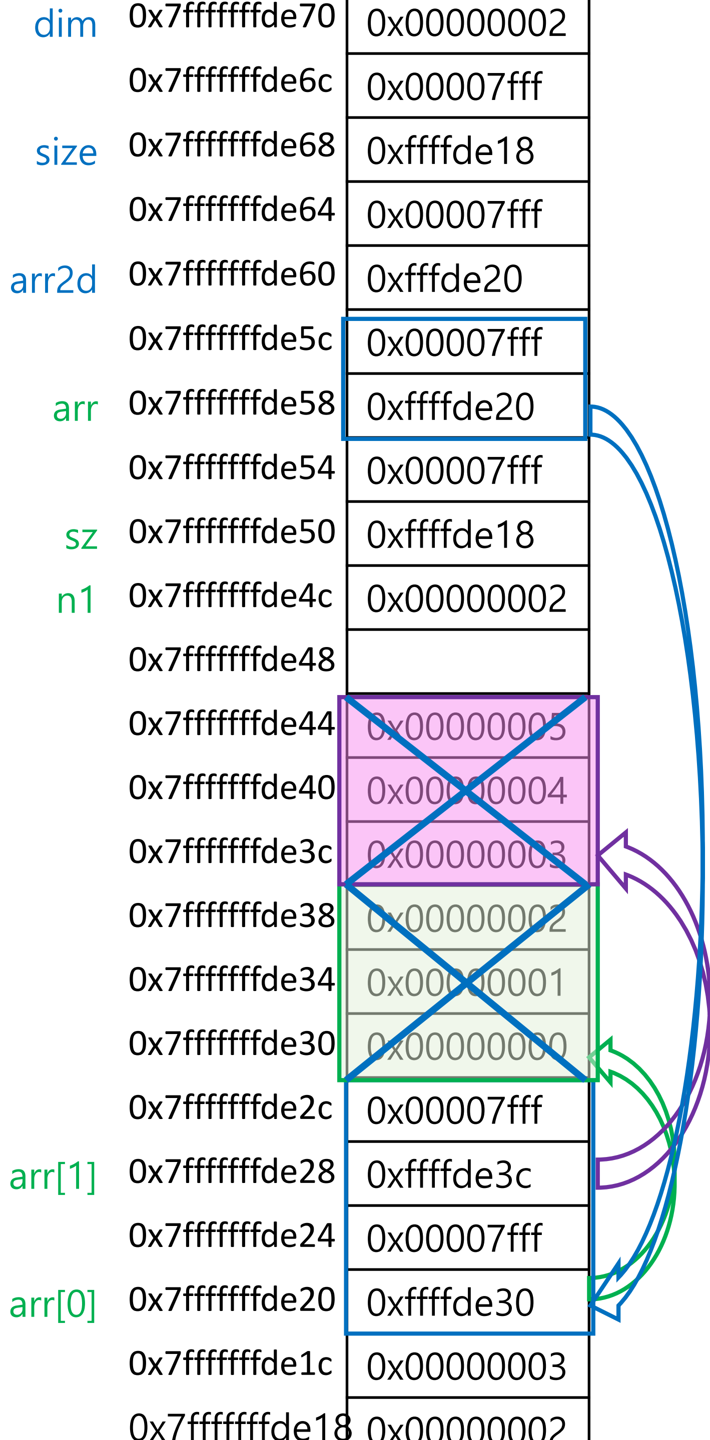
2차원 배열 destroyArray2D()



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

int **makeArray2D(int *sz);
void destroyArray2D(int **arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int **arr2d = NULL;
  arr2d = makeArray2D(size);
  for (int i=0; i<size[0]; i++)
     for (int j=0; j<size[1]; j++) arr2d[i][j] = i*size[1]+j;
  for (int i=0; i<size[0]; i++) {
     for (int j=0; j<size[1]; j++) cout <<  arr2d[i][j] << ' ';
     cout << endl;
  }
  destroyArray2D(arr2d, size);
  return 0;
}
```

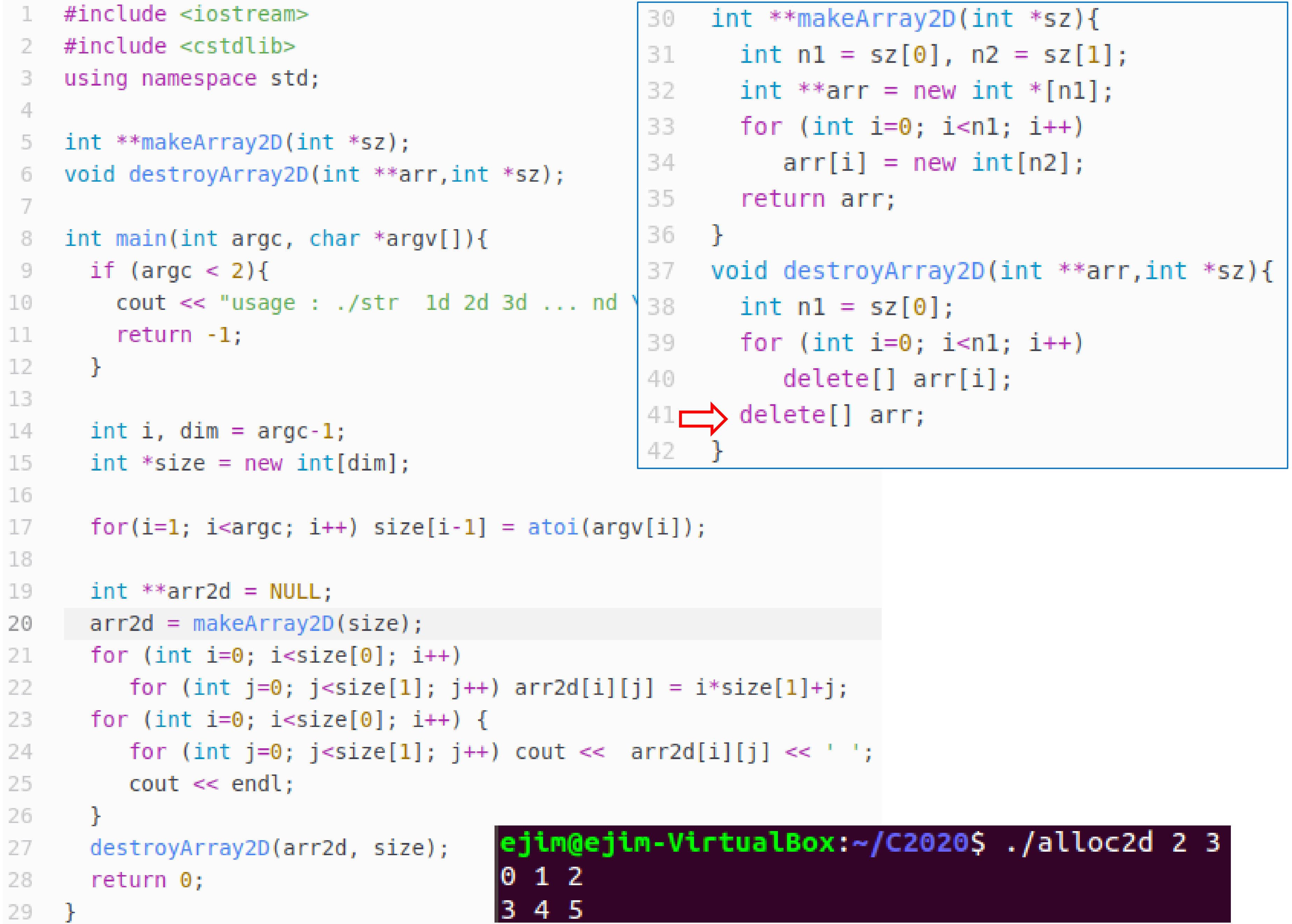
```cpp
int **makeArray2D(int *sz){
  int n1 = sz[0], n2 = sz[1];
  int **arr = new int *[n1];
  for (int i=0; i<n1; i++)
     arr[i] = new int[n2];
  return arr;
}
void destroyArray2D(int **arr,int *sz){
  int n1 = sz[0];
  for (int i=0; i<n1; i++)
     delete[] arr[i];
  delete[] arr;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./alloc2d 2 3
0 1 2
3 4 5
```



dim	0x7fffffffde70	0x00000002
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde18
	0x7fffffffde64	0x00007fff
arr2d	0x7fffffffde60	0xfffde20
	0x7fffffffde5c	0x00007fff
arr	0x7fffffffde58	0xffffde20
	0x7fffffffde54	0x00007fff
sz	0x7fffffffde50	0xffffde18
n1	0x7fffffffde4c	0x00000002
	0x7fffffffde48	
	0x7fffffffde44	~~0x00000005~~
	0x7fffffffde40	~~0x00000004~~
	0x7fffffffde3c	~~0x00000003~~
	0x7fffffffde38	~~0x00000002~~
	0x7fffffffde34	~~0x00000001~~
	0x7fffffffde30	~~0x00000000~~
	0x7fffffffde2c	0x00007fff
arr[1]	0x7fffffffde28	0xffffde3c
	0x7fffffffde24	0x00007fff
arr[0]	0x7fffffffde20	0xffffde30
	0x7fffffffde1c	0x00000003
	0x7fffffffde18	0x00000002

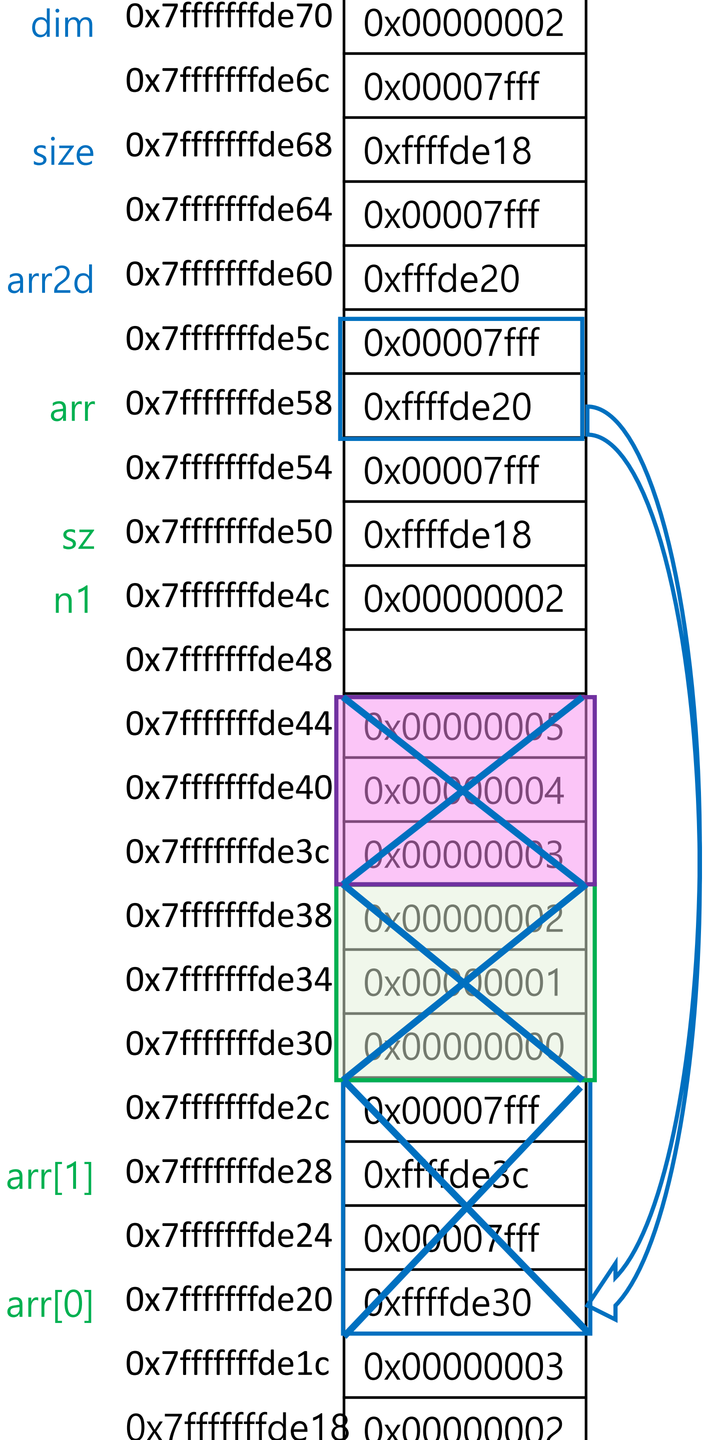
2차원 배열 destroyArray2D()



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

int **makeArray2D(int *sz);
void destroyArray2D(int **arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int **arr2d = NULL;
  arr2d = makeArray2D(size);
  for (int i=0; i<size[0]; i++)
     for (int j=0; j<size[1]; j++) arr2d[i][j] = i*size[1]+j;
  for (int i=0; i<size[0]; i++) {
     for (int j=0; j<size[1]; j++) cout <<  arr2d[i][j] << ' ';
     cout << endl;
  }
  destroyArray2D(arr2d, size);
  return 0;
}
```

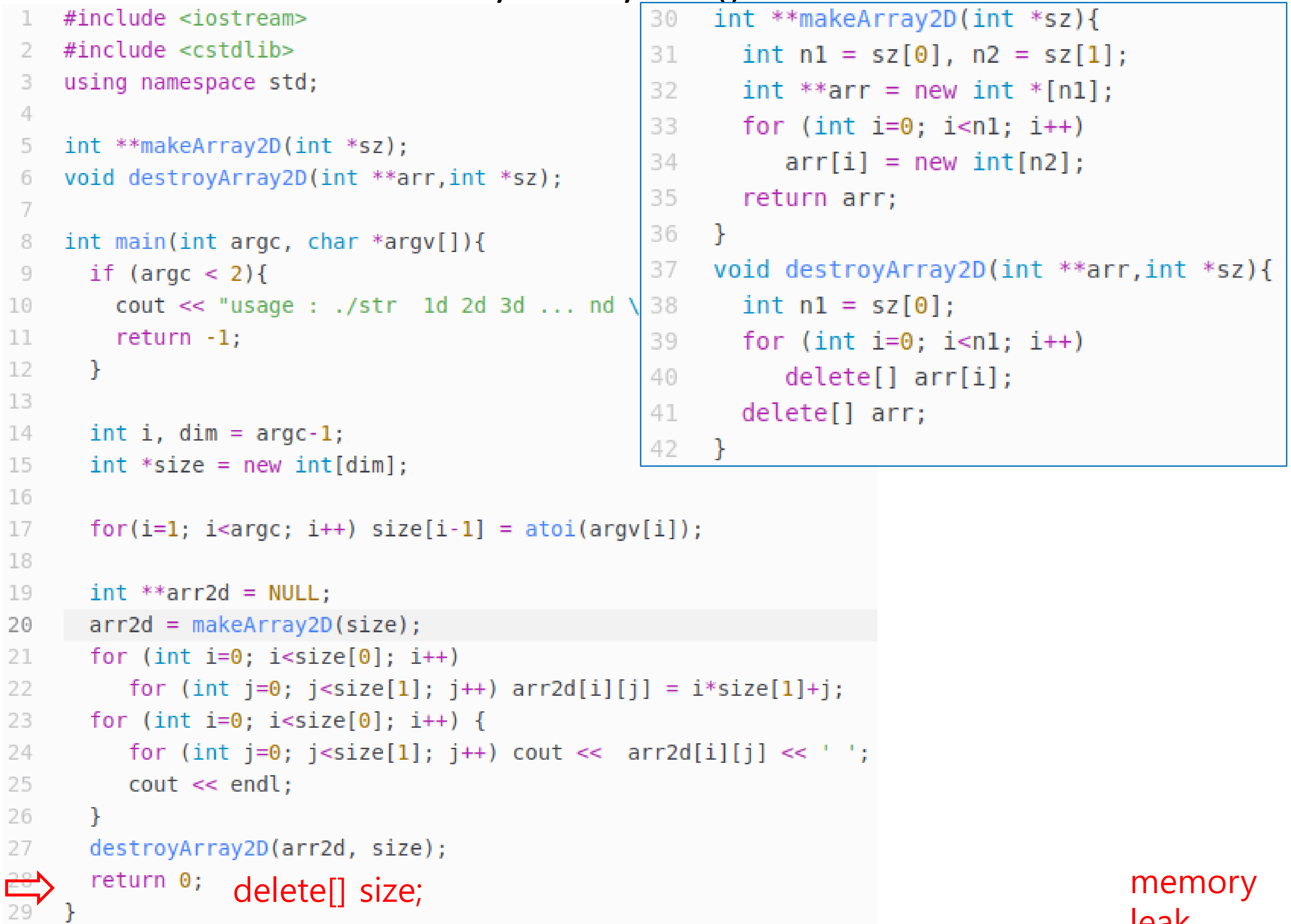
```
int **makeArray2D(int *sz){
  int n1 = sz[0], n2 = sz[1];
  int **arr = new int *[n1];
  for (int i=0; i<n1; i++)
     arr[i] = new int[n2];
  return arr;
}
void destroyArray2D(int **arr,int *sz){
  int n1 = sz[0];
  for (int i=0; i<n1; i++)
     delete[] arr[i];
  delete[] arr;
}
```

```
ejim@ejim-VirtualBox:~/C2020$ ./alloc2d 2 3
0 1 2
3 4 5
```



dim	0x7fffffffde70	0x00000002
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde18
	0x7fffffffde64	0x00007fff
arr2d	0x7fffffffde60	0xfffde20
	0x7fffffffde5c	0x00007fff
arr	0x7fffffffde58	0xffffde20
	0x7fffffffde54	0x00007fff
sz	0x7fffffffde50	0xffffde18
n1	0x7fffffffde4c	0x00000002
	0x7fffffffde48	
	0x7fffffffde44	0x00000005
	0x7fffffffde40	0x00000004
	0x7fffffffde3c	0x00000003
	0x7fffffffde38	0x00000002
	0x7fffffffde34	0x00000001
	0x7fffffffde30	0x00000000
	0x7fffffffde2c	0x00007fff
arr[1]	0x7fffffffde28	0xffffde3c
	0x7fffffffde24	0x00007fff
arr[0]	0x7fffffffde20	0xffffde30
	0x7fffffffde1c	0x00000003
	0x7fffffffde18	0x00000002

2차원 배열 destroyArray2D()

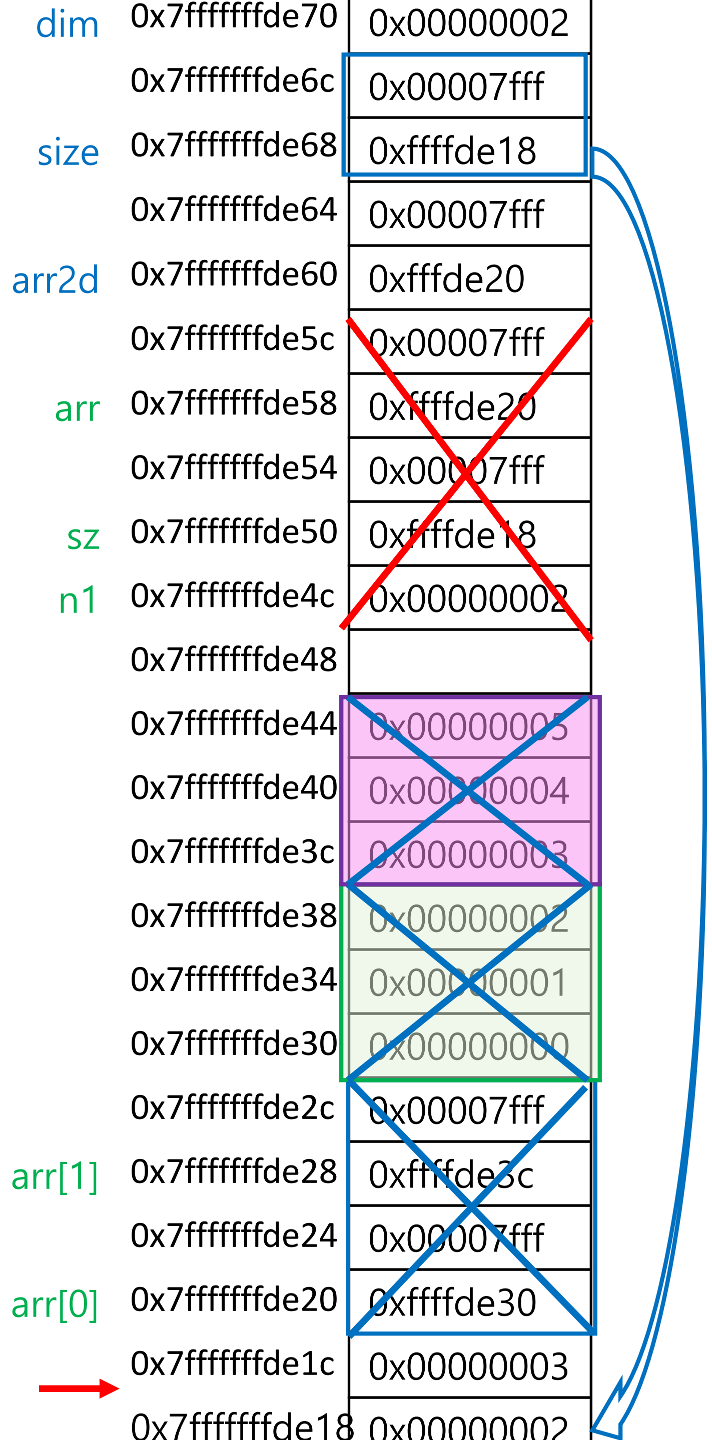


```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int **makeArray2D(int *sz);
void destroyArray2D(int **arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int **arr2d = NULL;
  arr2d = makeArray2D(size);
  for (int i=0; i<size[0]; i++)
     for (int j=0; j<size[1]; j++) arr2d[i][j] = i*size[1]+j;
  for (int i=0; i<size[0]; i++) {
     for (int j=0; j<size[1]; j++) cout <<  arr2d[i][j] << ' ';
     cout << endl;
  }
  destroyArray2D(arr2d, size);
  return 0;
}
```

delete[] size;

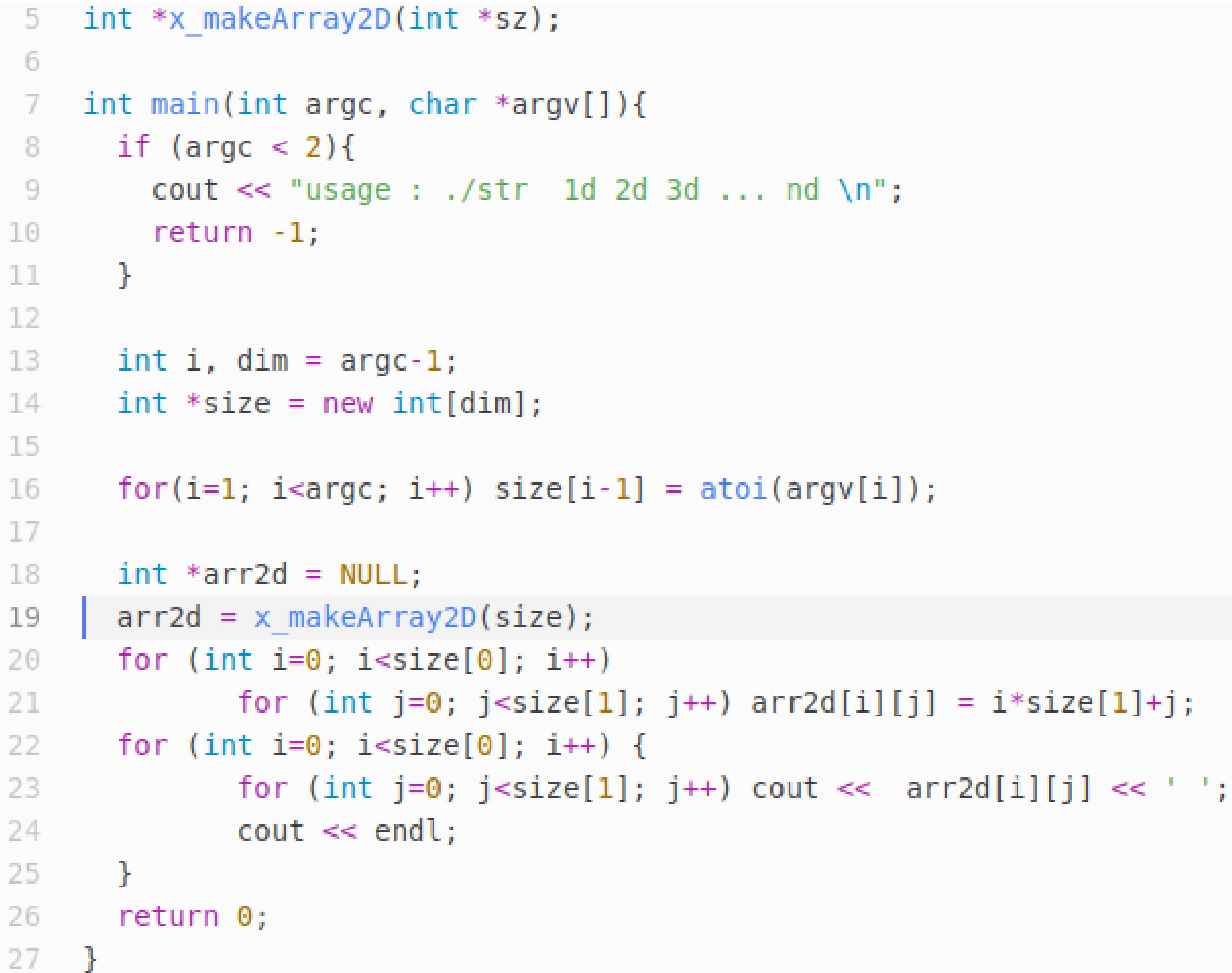
```cpp
int **makeArray2D(int *sz){
  int n1 = sz[0], n2 = sz[1];
  int **arr = new int *[n1];
  for (int i=0; i<n1; i++)
     arr[i] = new int[n2];
  return arr;
}
void destroyArray2D(int **arr,int *sz){
  int n1 = sz[0];
  for (int i=0; i<n1; i++)
     delete[] arr[i];
  delete[] arr;
}
```

memory leak

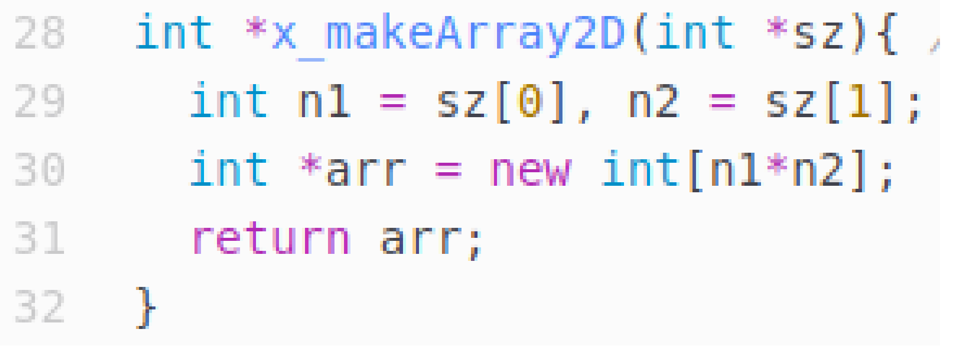


dim	0x7fffffffde70	0x00000002
	0x7fffffffde6c	0x00007fff
size	0x7fffffffde68	0xffffde18
	0x7fffffffde64	0x00007fff
arr2d	0x7fffffffde60	0xfffde20
	0x7fffffffde5c	~~0x00007fff~~
arr	0x7fffffffde58	~~0xffffde20~~
	0x7fffffffde54	~~0x00007fff~~
sz	0x7fffffffde50	~~0xffffde18~~
n1	0x7fffffffde4c	~~0x00000002~~
	0x7fffffffde48	
	0x7fffffffde44	~~0x00000005~~
	0x7fffffffde40	~~0x00000004~~
	0x7fffffffde3c	~~0x00000003~~
	0x7fffffffde38	~~0x00000002~~
	0x7fffffffde34	~~0x00000001~~
	0x7fffffffde30	~~0x00000000~~
	0x7fffffffde2c	~~0x00007fff~~
arr[1]	0x7fffffffde28	~~0xffffde3c~~
	0x7fffffffde24	~~0x00007fff~~
arr[0]	0x7fffffffde20	~~0xffffde30~~
	0x7fffffffde1c	0x00000003
	0x7fffffffde18	0x00000002

2차원 배열 : wrong example

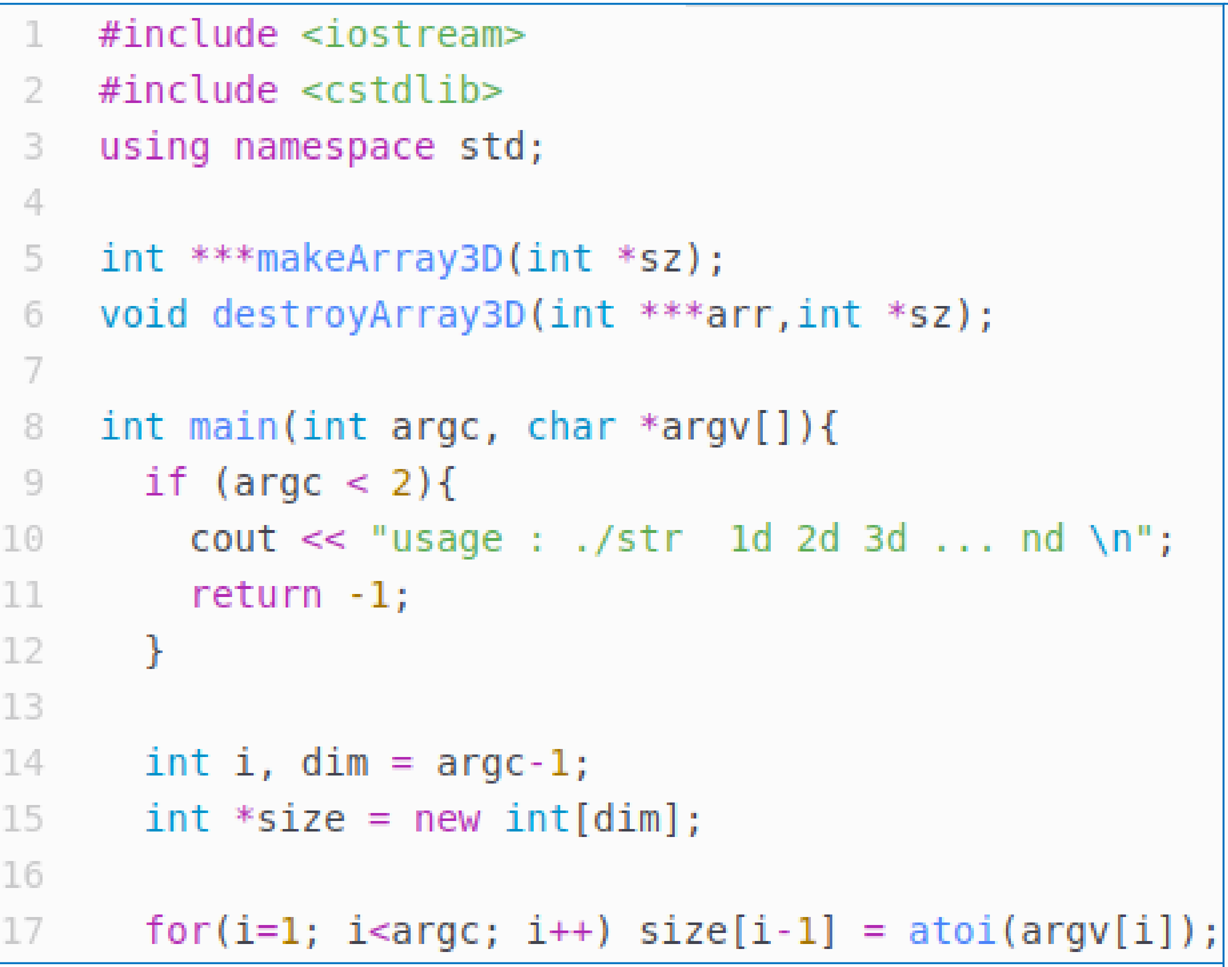


```
int *x_makeArray2D(int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);

  int *arr2d = NULL;
  arr2d = x_makeArray2D(size);
  for (int i=0; i<size[0]; i++)
         for (int j=0; j<size[1]; j++) arr2d[i][j] = i*size[1]+j;
  for (int i=0; i<size[0]; i++) {
         for (int j=0; j<size[1]; j++) cout <<  arr2d[i][j] << ' ';
         cout << endl;
  }
  return 0;
}
```

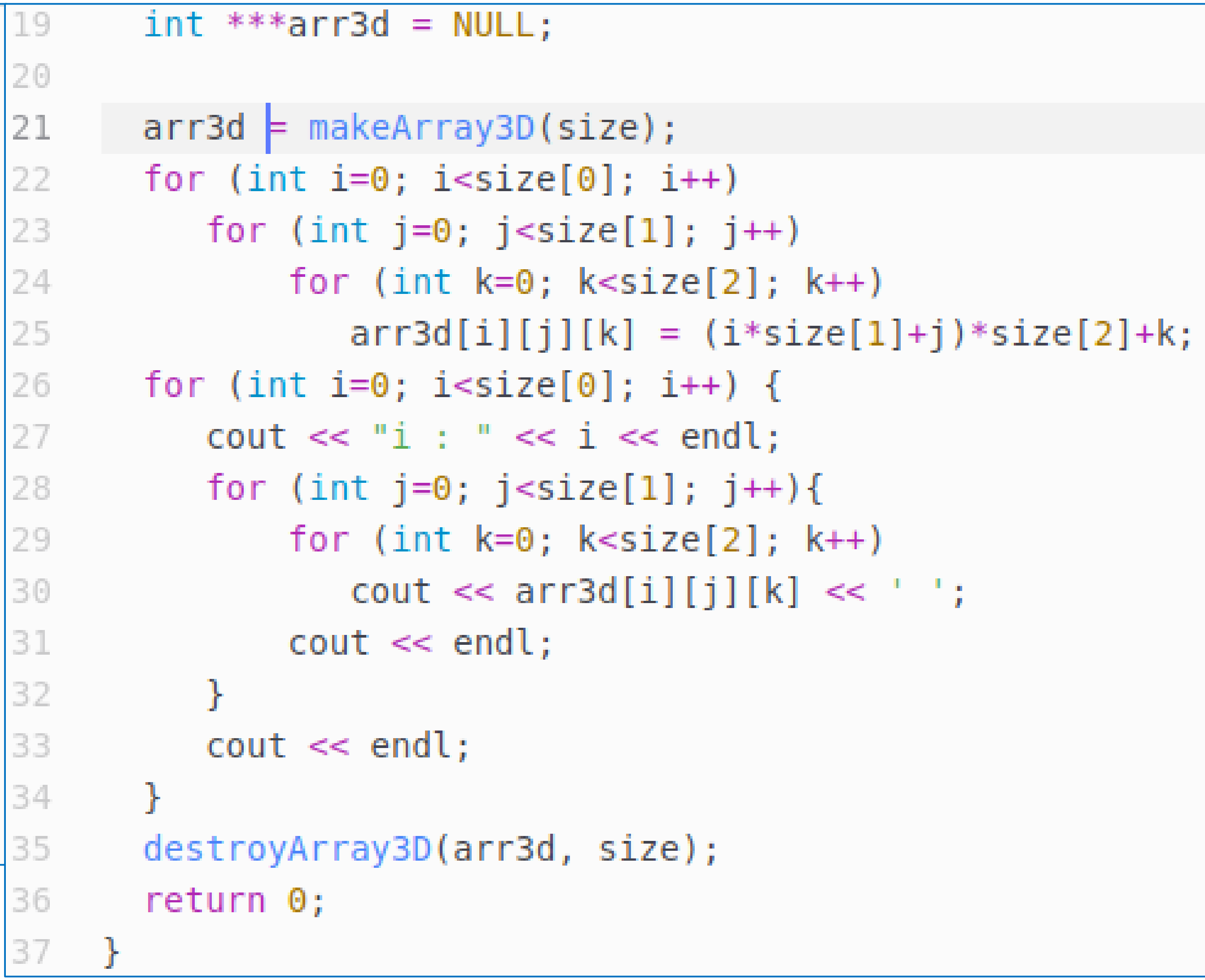


```
int *x_makeArray2D(int *sz){ /
  int n1 = sz[0], n2 = sz[1];
  int *arr = new int[n1*n2];
  return arr;
}
```

3차원 배열 만들어 반환하기 makeArray3D()

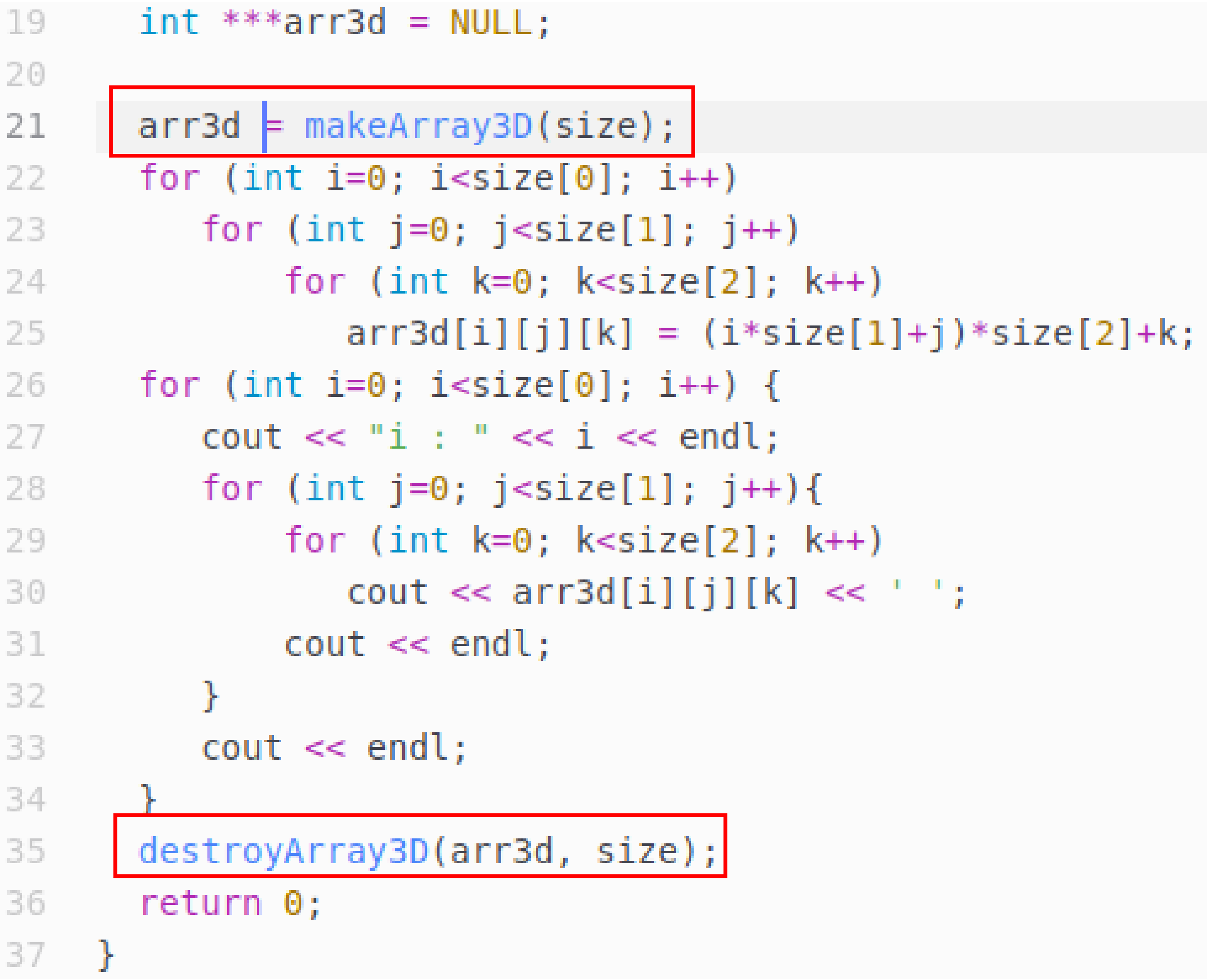


```
#include <iostream>
#include <cstdlib>
using namespace std;

int ***makeArray3D(int *sz);
void destroyArray3D(int ***arr,int *sz);

int main(int argc, char *argv[]){
  if (argc < 2){
    cout << "usage : ./str  1d 2d 3d ... nd \n";
    return -1;
  }

  int i, dim = argc-1;
  int *size = new int[dim];

  for(i=1; i<argc; i++) size[i-1] = atoi(argv[i]);
```

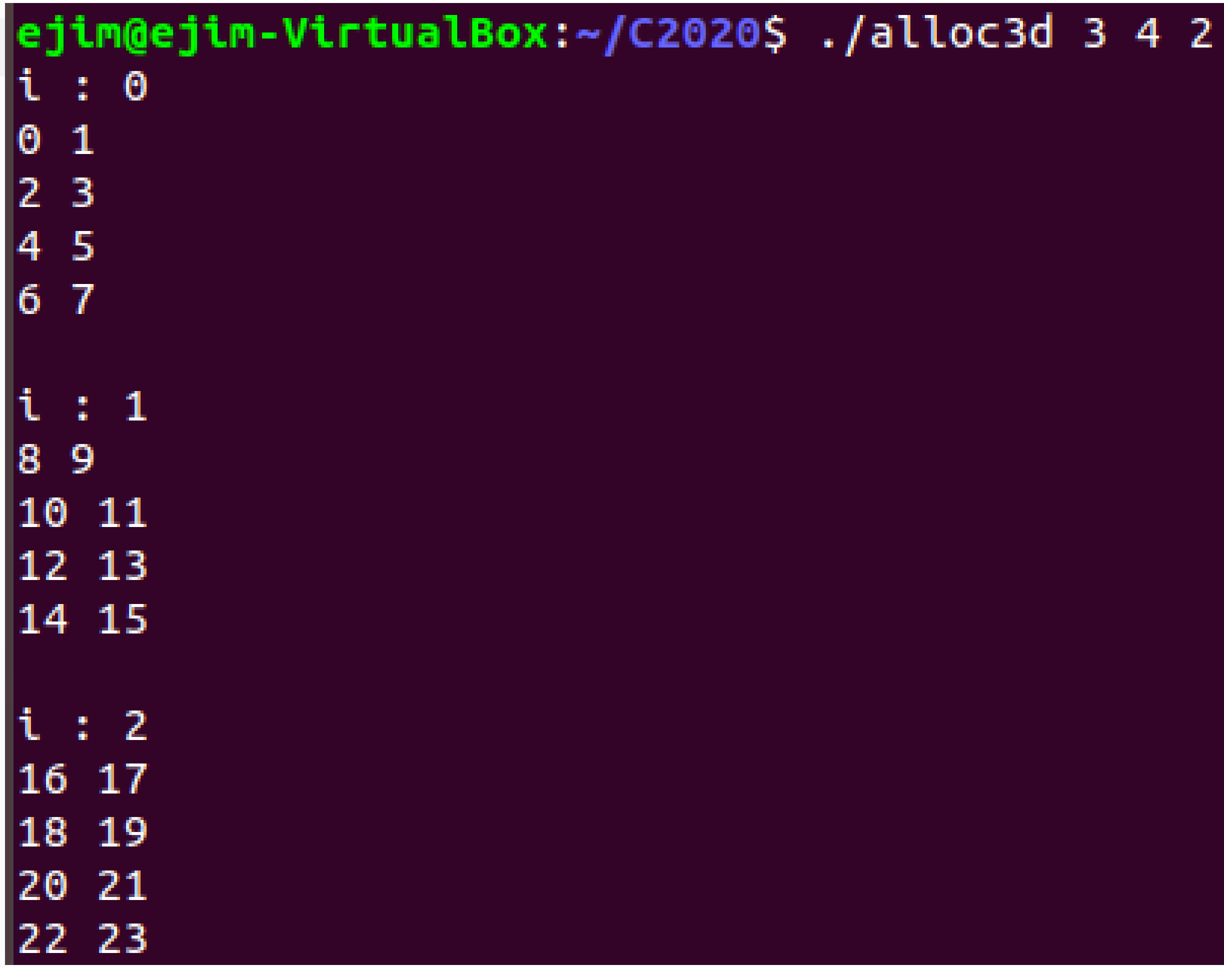


```
  int ***arr3d = NULL;

  arr3d = makeArray3D(size);
  for (int i=0; i<size[0]; i++)
     for (int j=0; j<size[1]; j++)
         for (int k=0; k<size[2]; k++)
            arr3d[i][j][k] = (i*size[1]+j)*size[2]+k;
  for (int i=0; i<size[0]; i++) {
     cout << "i : " << i << endl;
     for (int j=0; j<size[1]; j++){
         for (int k=0; k<size[2]; k++)
            cout << arr3d[i][j][k] << ' ';
         cout << endl;
     }
     cout << endl;
  }
  destroyArray3D(arr3d, size);
  return 0;
}
```

3차원 배열 만들어 반환하기 makeArray3D()



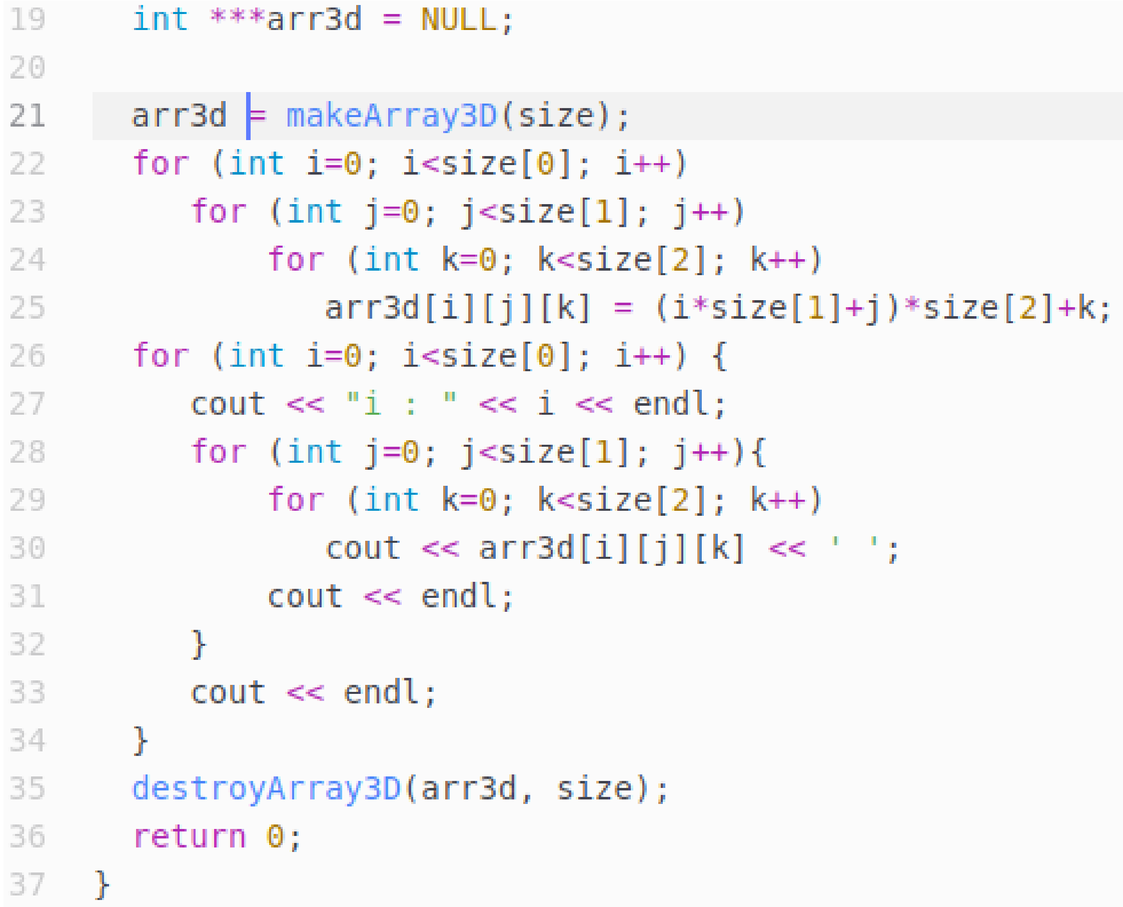
```
    int ***arr3d = NULL;

    arr3d = makeArray3D(size);
    for (int i=0; i<size[0]; i++)
        for (int j=0; j<size[1]; j++)
            for (int k=0; k<size[2]; k++)
                arr3d[i][j][k] = (i*size[1]+j)*size[2]+k;
    for (int i=0; i<size[0]; i++) {
        cout << "i : " << i << endl;
        for (int j=0; j<size[1]; j++){
            for (int k=0; k<size[2]; k++)
                cout << arr3d[i][j][k] << ' ';
            cout << endl;
        }
        cout << endl;
    }
    destroyArray3D(arr3d, size);
    return 0;
}
```



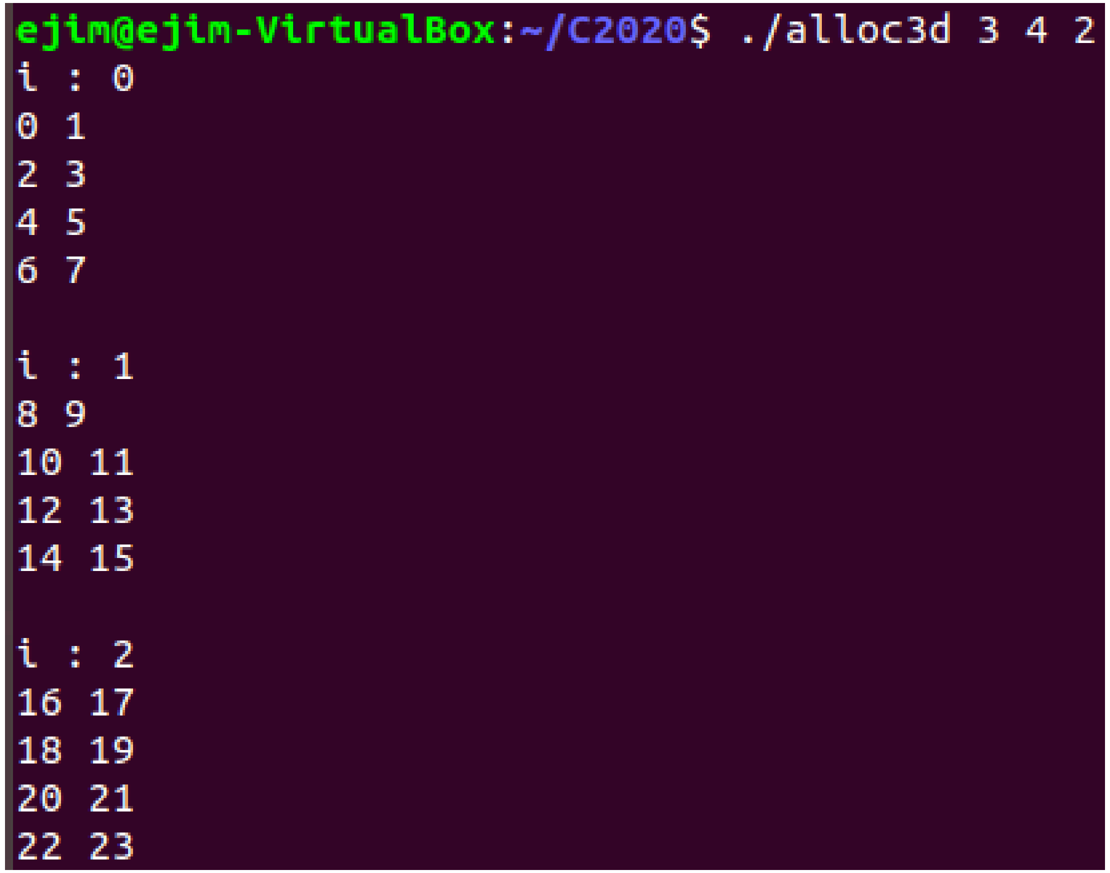
```
ejim@ejim-VirtualBox:~/C2020$ ./alloc3d 3 4 2
i : 0
0 1
2 3
4 5
6 7

i : 1
8 9
10 11
12 13
14 15

i : 2
16 17
18 19
20 21
22 23
```

실습

- 3차원 배열을 만들어서 반환하는 함수 makeArray3D() 와 3차원 함수를 heap 에서 제거하는 함수 destroyArray3D() 를 완성하라.
- 제출 방법 : ecampus 의 숙제 제출 text 내용을 e-campus editor 에 입력 (copy & paste 사용) screen capture 제출은 안 됨



```
  int ***arr3d = NULL;

  arr3d = makeArray3D(size);
  for (int i=0; i<size[0]; i++)
     for (int j=0; j<size[1]; j++)
         for (int k=0; k<size[2]; k++)
            arr3d[i][j][k] = (i*size[1]+j)*size[2]+k;
  for (int i=0; i<size[0]; i++) {
     cout << "i : " << i << endl;
     for (int j=0; j<size[1]; j++){
         for (int k=0; k<size[2]; k++)
            cout << arr3d[i][j][k] << ' ';
         cout << endl;
     }
     cout << endl;
  }
  destroyArray3D(arr3d, size);
  return 0;
}
```



```
ejim@ejim-VirtualBox:~/C2020$ ./alloc3d 3 4 2
i : 0
0 1
2 3
4 5
6 7

i : 1
8 9
10 11
12 13
14 15

i : 2
16 17
18 19
20 21
22 23
```