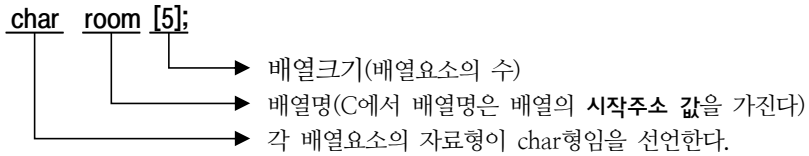
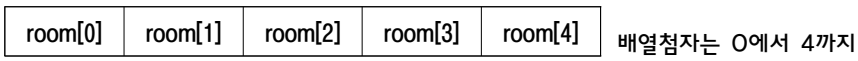
6. 배열(array)

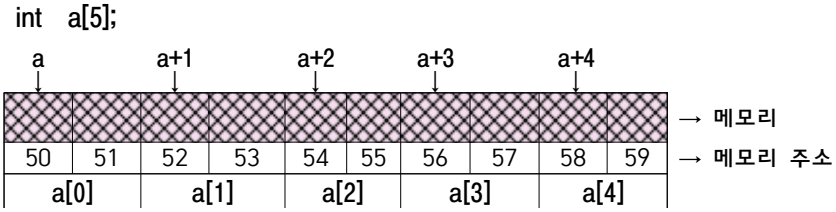
C에서 배열은 대괄호([ ])를 사용하여 선언한다.
즉, 배열 선언을 위한 특별한 **예약어는 없으며**, 다음과 같이 배열을 선언한다.



- 배열요소의 각 자료형은 char형이다.
- C에서 배열의 첨자는 0부터 시작된다.
- 위와 같이 선언하면 다음과 같이 5개의 배열요소가 확보된다.



◉ **배열 선언과 메모리 구조(16비트 컴퓨터인 경우)**



---

① 배열명은 메모리 중에 배열이 위치하게 되는 **시작주소**를 갖는다(a=50)
② 각 배열요소 하나하나는 메모리 2 byte씩 차지한다(int형으로 선언했으므로)
③ 따라서, 배열의 크기는 총 10 byte이다(2 * 5 = 10)
④ 배열명을 1씩 증가시키면
실제 메모리 증가분은 배열요소가 차지하는 바이트 수만큼 증가된다.(여기서는 2만큼)
⑤ 배열명은 배열의 시작주소 값을 가지는데,
1 증가시키면 메모리의 실제주소 값은 배열요소가 가지는 바이트 수만큼 증가된다.
(메모리의 실제주소 값이 증가되는 것은 포인터에서 다시 취급한다)

◈ 일차원 배열과 초기화

int a[5] = { 1, 2, 3, 4, 5 }; ⟶ 각 배열요소에 차례로 대입된다.

1	2	3	4	5
a[0]	a[1]	a[2]	a[3]	a[4]

↓예를 들어 설명하면

① int a[5] = {1, 2, }; → 초기화가 생략된 배열요소의 값은 0으로 초기화된다.

1	2	0	0	0
a[0]	a[1]	a[2]	a[3]	a[4]

② C에서 모든 배열요소를 **0으로 초기화**하는 방법

int a[5] = {0, }; → 초기값이 하나도 없으면 오류 발생(뒤에 콤마는 없어도 된다)

0	0	0	0	0
a[0]	a[1]	a[2]	a[3]	a[4]

◈ 배열의 배열(이차원 배열)

2행 3열의 배열구조에 다음과 같은 값이 대입되도록 배열을 초기화하는 방법이다.

a[2][3]

1 a[0][0]	2 a[0][1]	3 a[0][2]
4 a[1][0]	5 a[1][1]	6 a[1][2]

int a[2][3] = { 1, 2, 3, 4, 5, 6 }; → **행 우선**으로 배치된다.

int a[2][3] = { {1, 2, 3,}, {4, 5, 6} }; → 행 단위로 블록으로 묶었다.

[예] int a[ ][4] = { {5, 1, 4, 9},
{8, 2},
{6, 3, 7} };

5	1	4	9
8	2	0	0
6	3	7	0

→ 위와 같이 행 단위로 묶는 방식으로 초기화를 하면 초기값이 생략된 곳은 **0으로 채워진다.**

[Tip] 2차원 배열을 초기화할 때는 첫 번째 배열크기는 생략할 수 있다.
즉, int a[2][3]은 int a[ ][3]으로 기술해도 된다.

Escape sequence

Escape sequence는 문자를 표현하는 한 가지 방법으로
역 슬래쉬(\) 다음에 특정 기호를 기술하여 하나의 문자를 표현하는 것이다.

종 류	의 미	비 고
'\n'	커서를 다음 행으로 이동	New line
'\r'	커서를 현재 행의 첫 번째로 이동	Return
'\t'	커서를 다음 탭 위치로 이동	Tab
'\b'	커서를 앞으로 한 칸 이동	Back space
'\f'	인쇄용지를 1쪽 이동(Page skip)	Form feed
'\a'	소리 발생('삑'하고 경고음이 발생)	Beep
'\''	단일 따옴표(')를 지칭	
'\"'	이중 따옴표(")를 지칭	
'\\'	역 슬래쉬(₩)를 지칭	
'\수'	수에 해당하는 ASCⅡ문자(수는 8진수로 인식됨)	
'\x수'	수에 해당하는 ASCⅡ문자(수는 16진수로 인식됨)	

• C에서 문자는 8비트 기억공간에서 표현하며, 서로 다른 256가지의 종류가 있다.
• 한 문자를 단순히 단일 따옴표(' ')로 묶는 방법으로는 모든 문자 표현이 불가능하다.
• 그래서, Escape sequence라는 문자를 표현하는 또 다른 방법이 필요한 것이다.

• 다음은 C에서 대문자 'A' 를 출력하는 여러 가지 방법을 소개한 것이다.

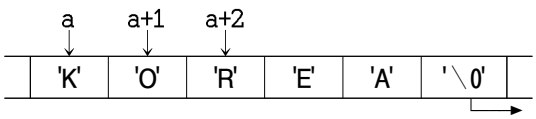
```
putchar ('A');
putchar (65);          → 수 '65' '은 10진수이다.
putchar ('\101');      → 수 '101'은 8진수이다.
putchar ('\x41');      → 수 '41'은 16진수이다.
putchar (0101);        → 수 '101'은 8진수이다.
putchar (0x41);        → 수 '41'은 16진수이다.
```

[Tip] 함수 putchar()는 하나의 문자를 출력하는 매크로 함수이다.
괄호 안의 인수는 하나의 문자를 표현하는 여러 가지 형태가 쓰일 수 있다.

◈ 배열과 문자열

예를 들어, 배열 a에 문자열 "KOREA"를 기억시키는 여러 방법을 살펴보자.

```
char a[ ] = {'K', 'O', 'R', 'E', 'A', '\0'};
char a[6] = "KOREA";
char a[ ] = "KOREA"; → 초기값이 있으므로 배열크기는 생략할 수 있다.
```



Null문자. 이는 프로그래머가 지정하지 않아도 문자열은 반드시 '\0'로 끝나야 하므로 컴파일러가 기억시켜 주게 된다.

---

① 배열명 a는 문자열 **시작주소** 값을 가진다.
② 문자열은 여러 개의 문자가 연결된 구조로 C에서는 반드시 '\0'으로 끝나야 한다.
'\0'은 Null문자라 부르며, C에서는 문자열의 끝을 판단하는데 이용한다.
③ 따라서, C에서 문자열을 처리할 때는 문자열의 길이에 대한 정보는 필요가 없다.
단지 문자열이 시작하는 메모리 주소값만을 기억하였다가
필요시 하나씩 꺼내어 처리하다가 '\0'를 만나면 종료하면 된다.
④ '\0'는 문자열 끝을 판단하는 것 이외에 여러 곳에서 사용된다.('\0'는 0이다)

[예]
```
void main()
{
  char a[] = "KOREA";
  printf("%s\n", a);        //문자열 출력(배열명 a는 문자열 시작주소 값을 가진다.
  printf("%s\n", a+2);      //a+2는 문자 R이 기억된 주소 값이다.
}
```
실행결과 : KOREA
REA → a+2이므로

Tip	함수 printf()는 변환기호 %s가 있을 때는 %s에 대응하는 인수가 가지는 주소부터 메모리에 수록된 내용을 차례로 꺼내어 문자로 출력시킨다. '\0'이 있으면 중지한다.

◈ 여러 개의 문자열 취급

• 여러 개의 문자열을 한꺼번에 배열에 기억시키려면 배열의 배열을 이용해야 한다.
• 문자열 "RED", "WHITE", "BLUE"를 배열 color에 기억시키는 방법을 살펴보자.

'R'	'E'	'D'	'\0'	'\0'	'\0'
'W'	'H'	'I'	'T'	'E'	'\0'
'B'	'L'	'U'	'E'	'\0'	'\0'

→ 열의 크기는 '가장 긴 문자길이+1'이어야 한다.
('\0'가 수록되어야 하므로)

```
char color[3][6] = { "RED", "WHITE", "BLUE" };
```

6은 생략할 수 없다
3은 생략할 수 있다

---

① 첫 번째 첨자는 생략할 수 있으나 두 번째는 생략할 수 없다.
② 배열의 열 크기는 '가장 긴 문자길이 + 1'이어야 한다.
③ 끝에 '\0'을 기억시킬 배열요소가 없으면 '\0'가 문자열 끝에 들어가지 않는다.
→ 완전한 문자열이 될 수 없다.
④ 배열의 자료형은 char형으로 한다. 배열 각 요소에 하나의 문자를 기억시키므로

◉ 여러 개의 문자열을 다루는 프로그램

```
void  main()
{
      char a[][6] = {"RED", "WHITE", "BLUE"};
      printf("%s\n", a[0]);                    //문자열 출력
      printf("%s\n", a[1]);
      printf("%s\n", a[2]);
 }
```

실행 결과 : RED
WHITE
BLUE

↓ 위의 예처럼 배열의 배열을 선언하고 초기화했을 때

• 각 행의 시작번지 값은 a[0], a[1], a[2]에 각각 기억된다.
• 즉, a[1]에는 문자열 "WHITE"의 "W"가 기억된 메모리 번지가 기억된다.

기출문제 분석

1. 다음 C 프로그램의 출력 결과는? [2019년 국가 9급]

```
#include <stdio.h>
int main()
{
  char msg[50] = "Hello World!! Good Luck!";
  int i = 2, number = 0;
  while(msg[i] != '!') {
    if(msg[i] == 'a' || msg[i] == 'e' || msg[i] == 'i' || msg[i] == 'o' || msg[i] == 'u')
      number++;
    i++;
  }
  printf("%d", number);
  return 0;
}
```

① 2 ② 3 ③ 5 ④ 6

♤ C 프로그램의 출력 결과

• msg[50] = "Hello World!! Good Luck!"

H	e	l	l	o		W	o	r	l	d	!	!		G	o	o	d		L	u	c	k	!	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

↓ 탐색 시작위치(i=2이므로)

└→ 탐색 종료위치(while문 탈출)

```
int i = 2, number = 0;
while(msg[i] != '!')     //문자 '!'를 만나면 조건이 거짓이 되어 반복문 탈출
{
  if(msg[i] == 'a' || msg[i] == 'e' || msg[i] == 'i' || msg[i] == 'o' || msg[i] == 'u')
      number++;
      i++;
}
```

↓ | l | l | o | | W | o | r | l | d | 에서 조건을 만족하는 문자는 2개('o')

number = 2

정답 : ①

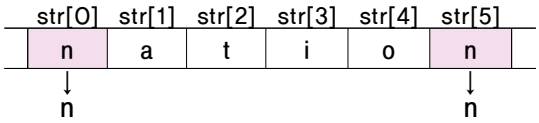
2. 다음 C 프로그램을 실행한 결과로 옳은 것은? [2015년 서울 9급]

```
-----------------------------
 void main(void){
   int i;
   char ch;
   char str[7] = "nation";
   for(i = 0; i < 4; i++){
     ch = str[5-i];
     str[5-i] = str[i];
     str[i] = ch;
   }
   printf("%s \n", str);
 }
-----------------------------
```

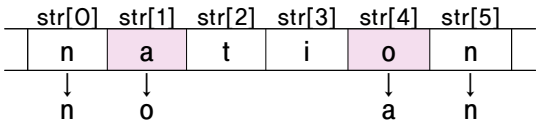
① nanoit ② nation ③ noitan ④ notian

♤ **메모리 구조 변화 – 매우 지저분하게 출제된 문제**

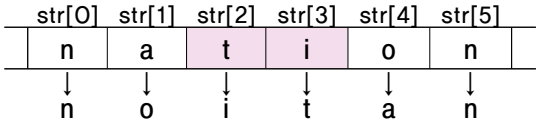
i = 0 일 때	ch = str[5] = n str[5] = str[0] = n str[0] = ch = n



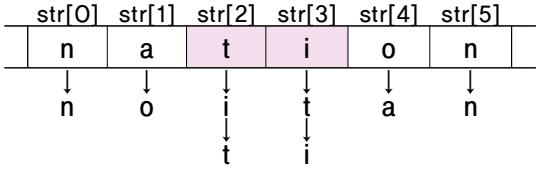
i = 1 일 때	ch = str[4] = o str[4] = str[1] = a str[1] = ch = o



i = 2 일 때	ch = str[3] = i str[3] = str[2] = t str[2] = ch = i



i = 3 일 때	ch = str[2] = i str[2] = str[3] = t str[3] = ch = i



◉ **이런 유형의 문제를 푸는 방법**

- 이런 유형의 문제는 변수 값이 변화되는 것을 하나하나 따져서 풀 수밖에 없다.
- 풀어 본 결과는, **양끝**에서 안으로 가면서 **좌우**의 문자를 **맞교환**하게 된다.
- 프로그램을 분석하지 않고서 이런 결과가 된다는 것을 알 수 없다.

정답 : ④