

-Part2-

제1장 1차원 배열이란 무엇인가

학습목차

1.1 1차원 배열이란

1.2 1차원 배열의 주소와 값의 참조

1.1 1차원 배열이란

1.1 1차원 배열이란 (1/12)

▶ 배열이란

- ✓ 같은 자료형을 가진 연속된 메모리 공간으로 이루어진 자료구조
- ✓ 같은 자료형을 가진 변수들이 여러 개 필요할 때 사용
- ✓ 많은 양의 데이터를 처리할 때 유용

```
#include <stdio.h>
int main(void)
{
    // int형 변수 30개
    int student1, student2, ... , student30;
    ...
    return 0;
}
```



```
#include <stdio.h>
int main(void)
{
    // int형 배열
    int student[30];
    ...
    return 0;
}

//코드의 길이가 짧아짐
//가독성 향상
```

1.1 1차원 배열이란 (2/12)

▶ 배열의 선언

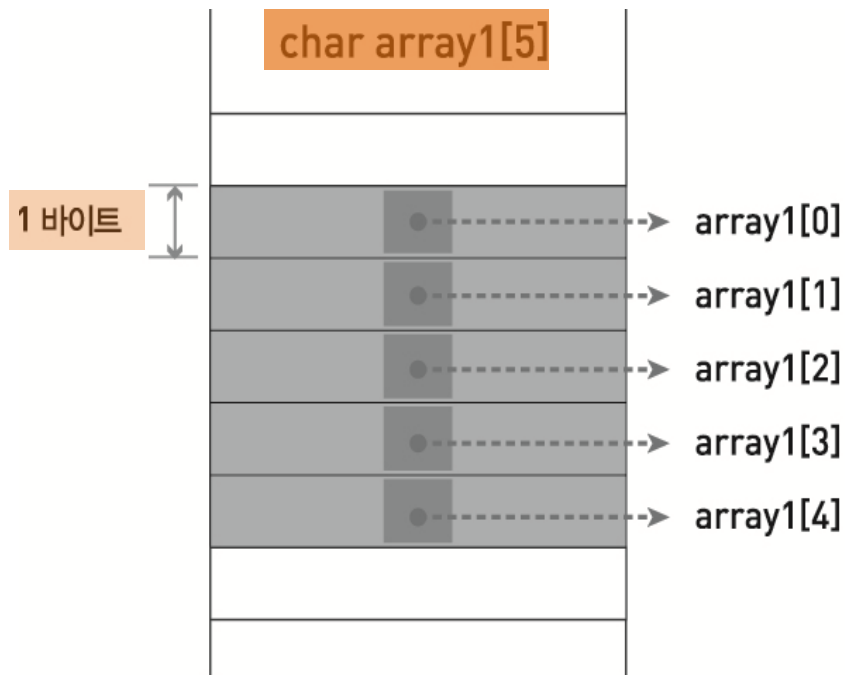
- ✓ 자료형: 배열의 자료형을 지정
- ✓ 배열 이름: 변수 이름과 마찬가지로 배열을 구분하는 이름
- ✓ 배열 길이: 배열 요소의 총 길이(10개의 변수를 배열로 구성)

자료형	배열 이름	[배열 길이]
↓	↓	↓
int	array	[10] ;

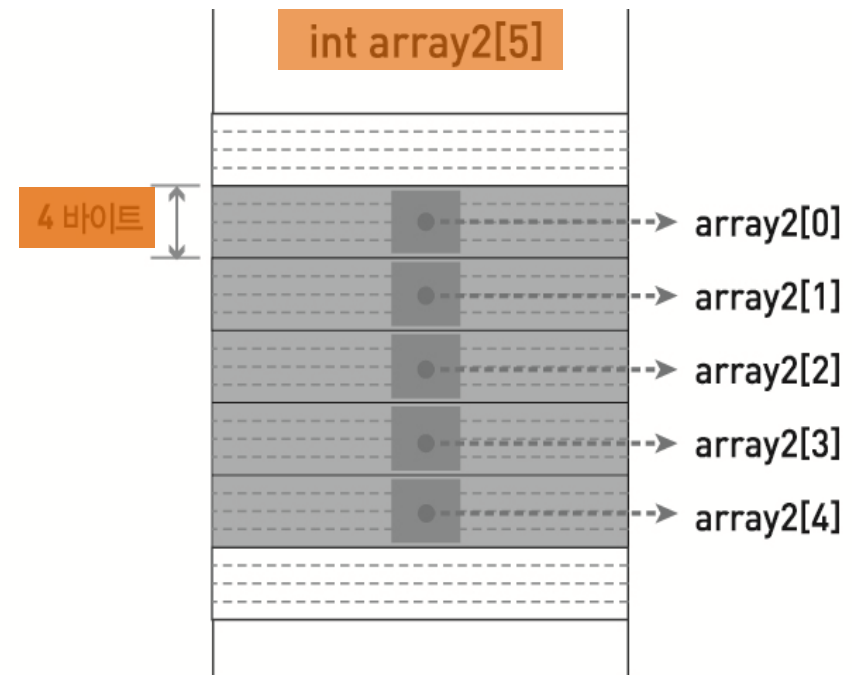
1.1 1차원 배열이란 (3/12)

▶ 배열 요소의 위치

✓ 0부터 시작



① 총 5바이트 크기의 연속된 메모리 공간을 할당하며 배열 요소는 0부터 시작



② 총 20바이트 크기의 연속된 메모리 공간을 할당하며 배열 요소는 0부터 시작

1.1 1차원 배열이란 (4/12)---[1-1.c 실습]

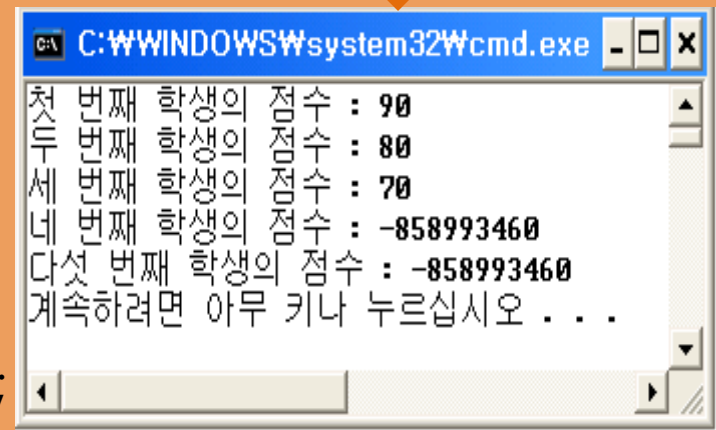
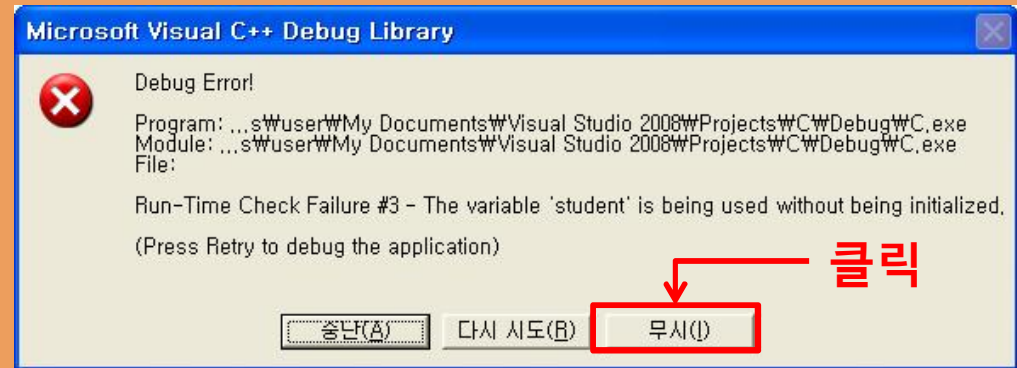
```
#include <stdio.h>
int main(void)
{
    int student[5];
```

```
    student[0] = 90;
    student[1] = 80;
    student[2] = 70;
```

```
    printf("첫 번째 학생의 점수 : %d \n", student[0]);
    printf("두 번째 학생의 점수 : %d \n", student[1]);
    printf("세 번째 학생의 점수 : %d \n", student[2]);
    printf("네 번째 학생의 점수 : %d \n", student[3]);
    printf("다섯 번째 학생의 점수 : %d \n", student[4]);
```

```
    return 0;
```

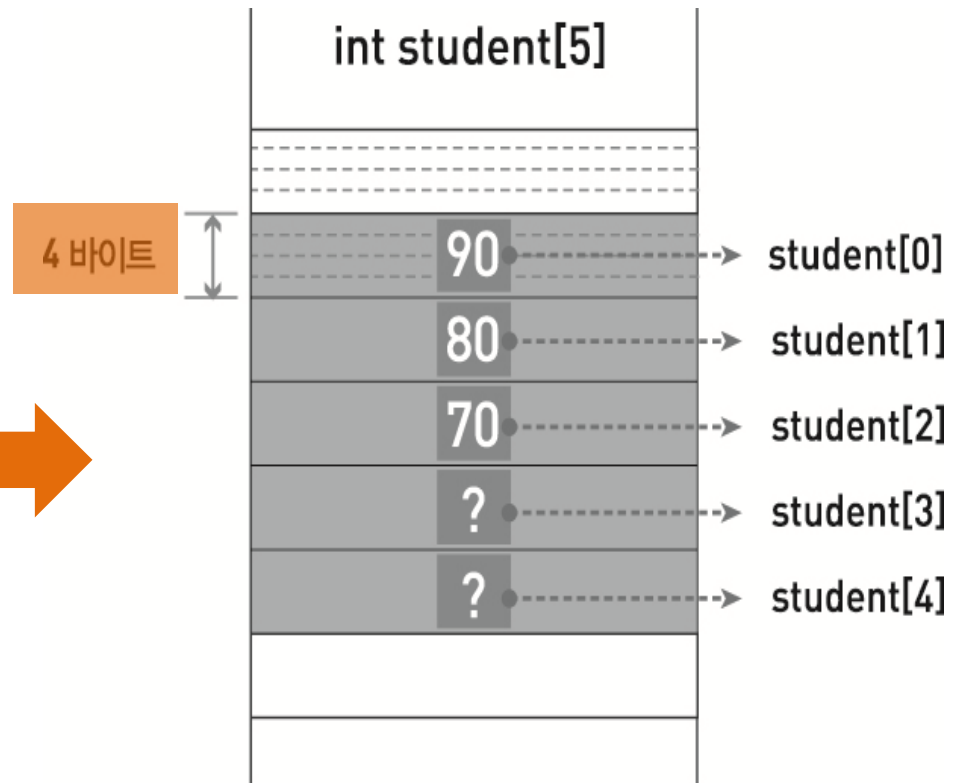
```
}
```



1.1 1차원 배열이란 (5/12)---[1-1.c 분석]

```
int student[5];
```

```
student[0] = 90;  
student[1] = 80;  
student[2] = 70;
```



총 20바이트 크기의 연속된 메모리 공간을
할당하며 배열 요소는 0부터 시작

1.1 1차원 배열이란 (6/12)---[1-2.c 실습]

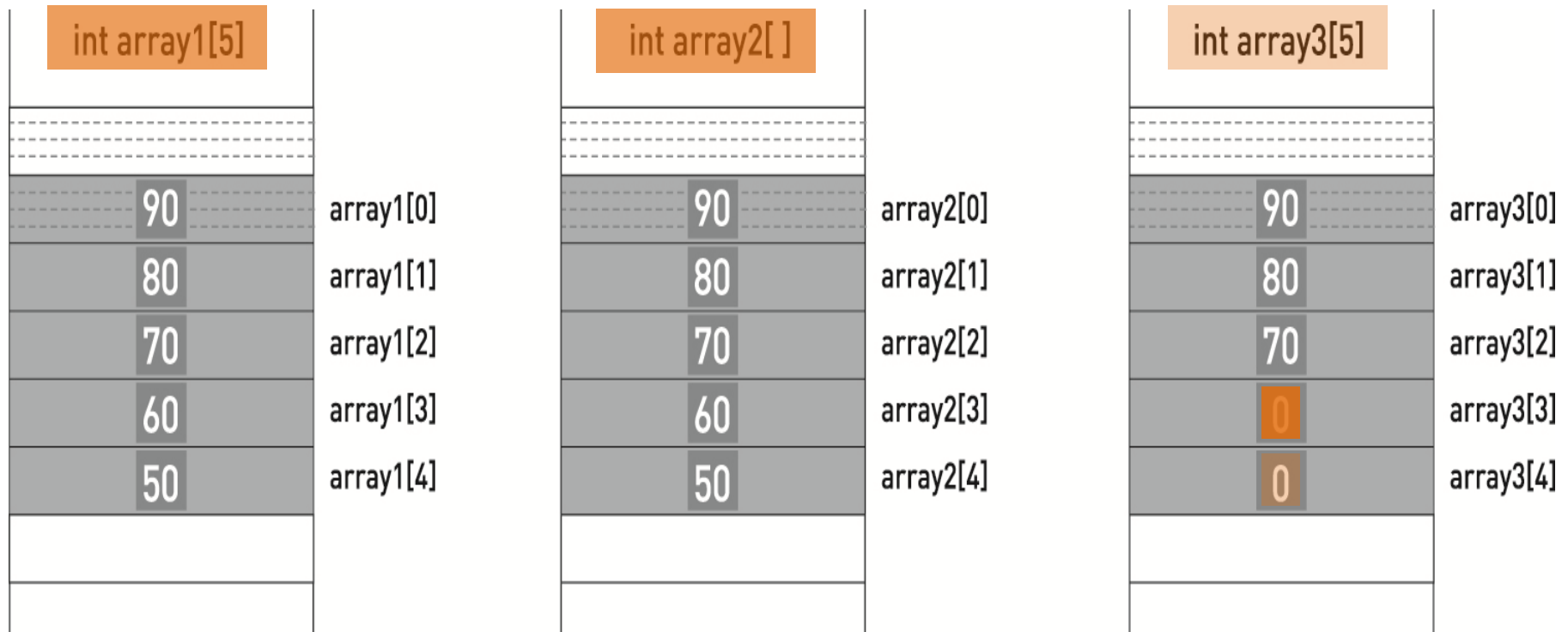
```
#include <stdio.h>
int main(void)
{
    int array1[5] = {90,80,70,60,50}; // 배열 선언 과 동시에 저장(초기화)
    int array2[ ] = {90,80,70,60,50};
    int array3[5] = {90,80,70};

    printf("%d %d %d %d %d \n",
           array1[0],array1[1],array1[2],array1[3],array1[4]);
    printf("%d %d %d %d %d \n",
           array2[0],array2[1],array2[2],array2[3],array2[4]);
    printf("%d %d %d %d %d \n",
           array3[0],array3[1],array3[2],array3[3],array3[4]);

    return 0;
}
```

1.1 1차원 배열이란 (7/12)---[1-2.c 분석]

```
int array1[5] = {90,80,70,60,50};
int array2[ ] = {90,80,70,60,50};
int array3[5] = {90,80,70};
```

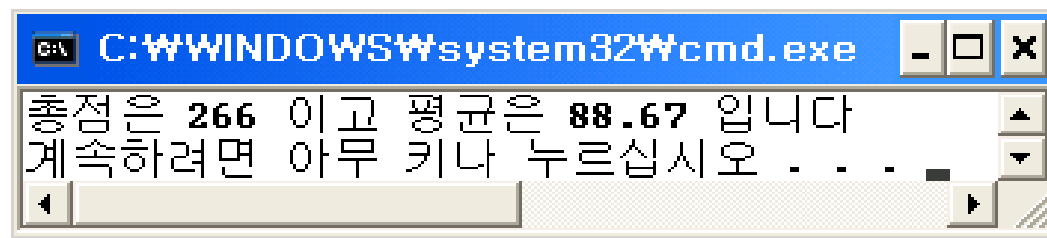


1.1 1차원 배열이란 (8/12)---[1-3.c 실습]

```
#include <stdio.h>
int main(void)
{
    int array[3]={87,99,80};
    int total=0;

    total=array[0]+array[1]+array[2];
    printf("총점은 %d 이고 ", total);
    printf("평균은 %.2lf 입니다\n", (double)total/3);

    return 0;
}
```



1.1 1차원 배열이란 (9/12)---[1-4.c 실습]

```
#include<stdio.h>
int main(void)
{
    int array[3]={87,65,78};
    int i, total=0;

    for(i=0; i<3; i++)
    {
        total = total + array[i];
        printf("배열의 요소 array[%d]의 값: %d \n", i, array[i]);

    }

    printf("총점은 %d 이고 ", total);
    printf("평균은 %.2lf 입니다\n", (double)total/3);

    return 0;
}
```

1.1 1차원 배열이란 (10/12)

▶ 배열 선언 시 주의할 점

- ① '배열 요소는 **0부터 시작**한다.'

```
#include <stdio.h>
int main(void)
{
    int array[2]; // 배열 길이는 2

    array[0]=1; // 배열 요소는 0부터 시작
    array[1]=2;
    array[2]=3; // 에러

    return 0;
}
```

1.1 1차원 배열이란 (11/12)

▶ 배열 선언 시 주의할 점

- ② '배열 초기화를 중괄호로 할 때 배열의 선언과 초기화가 개별적으로 이루어져서는 안 된다.'

```
#include <stdio.h>
int main(void)
{
    int array1[3]={10, 20, 30}; // 정상적인 초기화 방법
    int array2[3];
    array2={10, 20, 30};      // 에러 발생

    return 0;
}
```

1.1 1차원 배열이란 (12/12)

▶ 배열 선언 시 주의할 점

- ③ '배열 길이'를 변수로 설정하면 안 된다!
- '배열의 길이는 상수로 설정'해야 한다!

```
#include <stdio.h>
#define MAX 10          // 심볼릭 상수 선언
int main(void)
{
    int a=3;
    const int SIZE=20;   // 심볼릭 상수 선언
    int array1[MAX];     // 정상: 배열 길이를 심볼릭 상수로 설정
    int array2[SIZE];    // 정상: 배열 길이를 심볼릭 상수로 설정
    int array3[a];       // 에러: 배열 길이를 변수로 설정

    return 0;
}
```

1.2 1차원 배열의 주소와 값의 참조

1.2 1차원 배열의 주소와 값의 참조 (1/12)

▶ '**&**는 주소를 참조하는 연산자이다.'

✓ & 연산자(주소 연산자)

• 메모리 공간의 **주소**를 표현

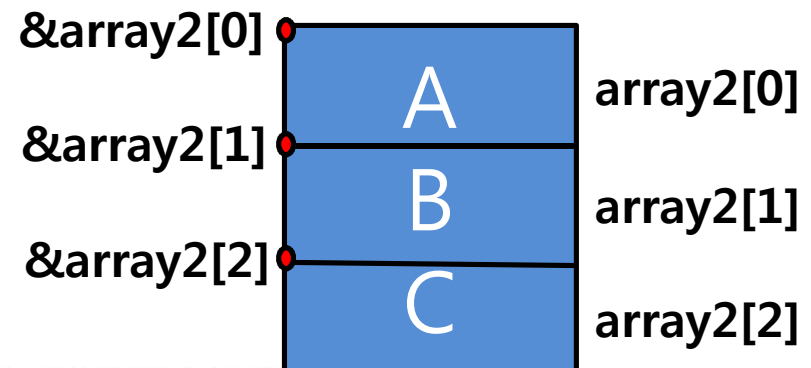
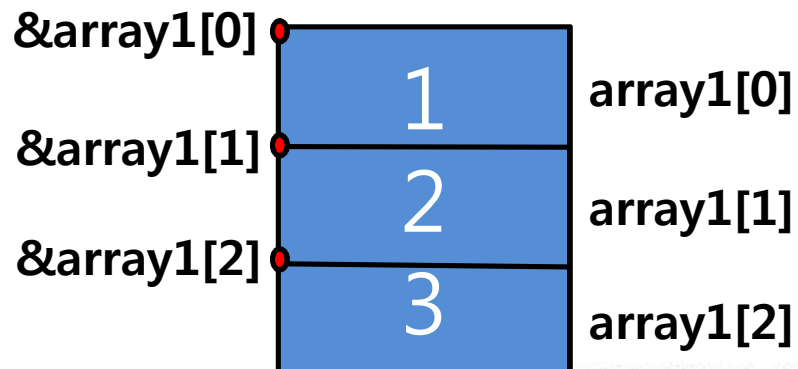
변수의 주소 표현	배열의 주소 표현
사용법: & 변수이름	사용법: & 배열 요소의 위치
<pre>int a=10, b=20; printf("%x \n", &a); // a의 주소 printf("%x \n", &b); // b의 주소</pre>	<pre>int array[2] = {10, 20}; printf("%x \n", &array[0]); // array[0]의 주소 printf("%x \n", &array[1]); // array[1]의 주소</pre>

1.2 1차원 배열의 주소와 값의 참조 (2/12)---[1-5.c 실습]

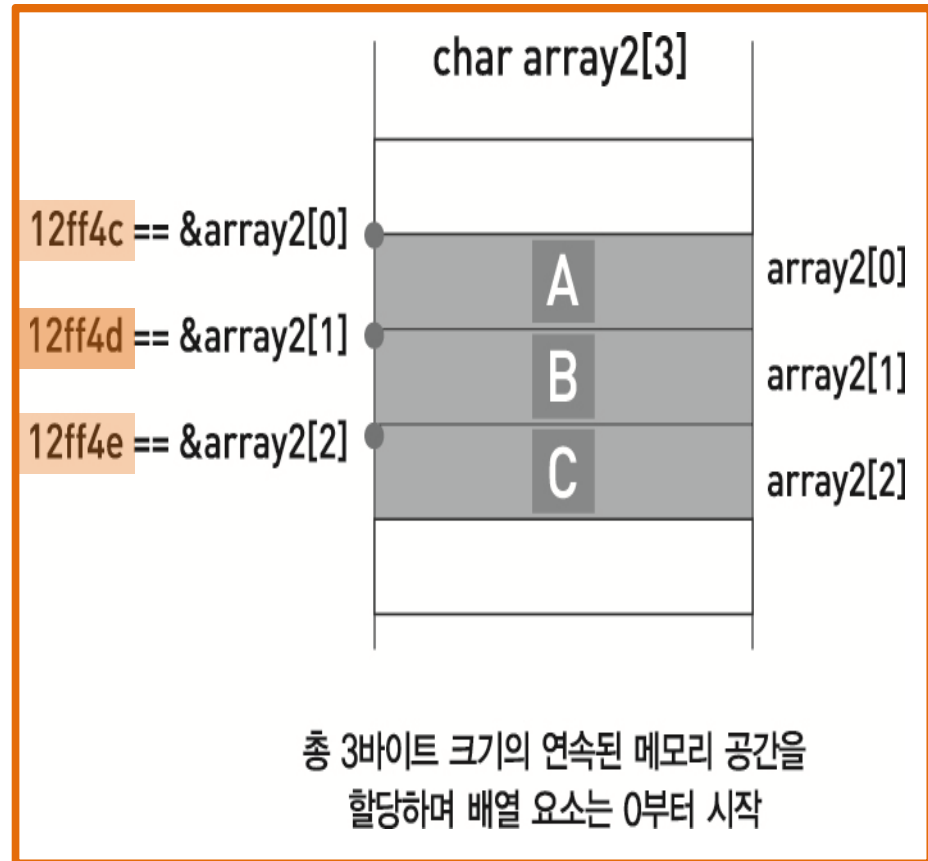
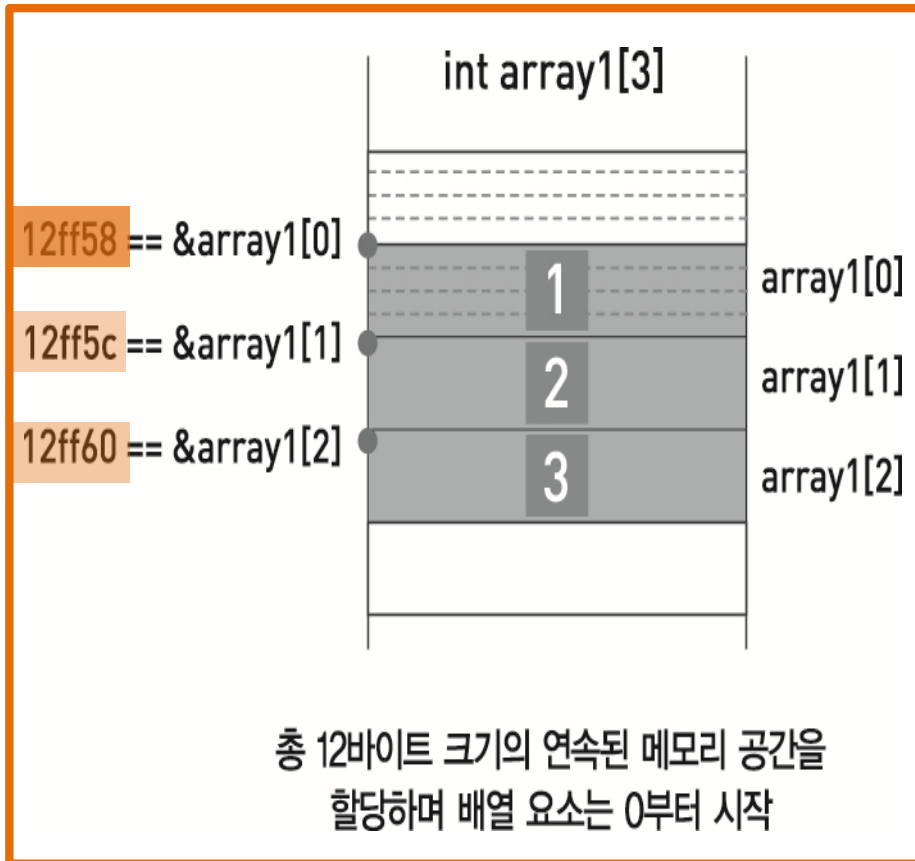
```
#include <stdio.h>
int main(void)
{
    int  array1[3] = {1,2,3};
    char array2[3] = {'A','B','C'};

    printf("%x %x %x \n", &array1[0], &array1[1], &array1[2]);
    printf("%x %x %x \n", &array2[0], &array2[1], &array2[2]);

    return 0;
}
```



1.2 1차원 배열의 주소와 값의 참조 (3/12)---[1-5.c 분석]

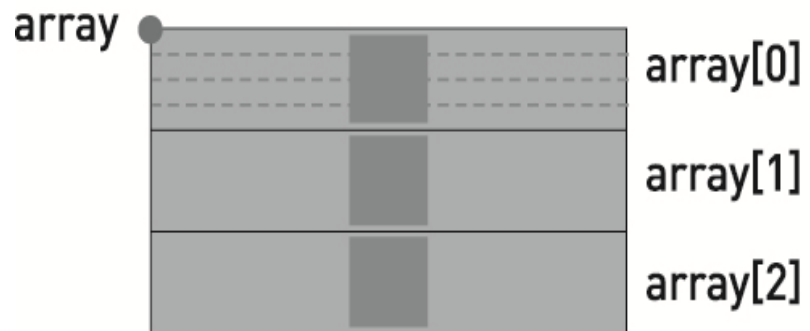


'주소를 나타내는 숫자 상수는 운영체제나 개발 SW 마다 다르다.'

1.2 1차원 배열의 주소와 값의 참조 (4/12)

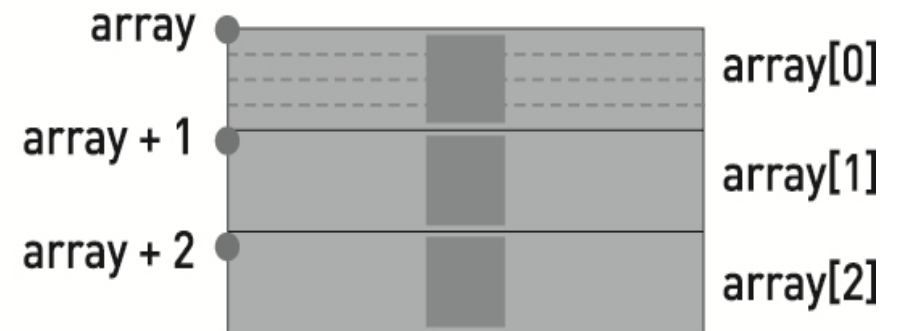
▶ '배열 이름은 배열의 시작 주소이다.'

int array[3]



배열 이름 array는 배열의 시작 주소

int array[3]



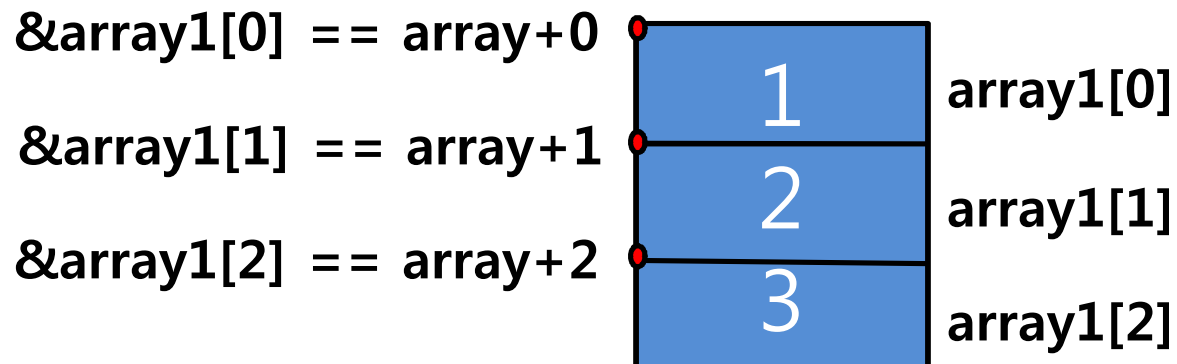
배열의 시작 주소를 기준으로
배열 요소의 개별 주소를 참조

1.2 1차원 배열의 주소와 값의 참조 (5/12)---[1-6.c 실습]

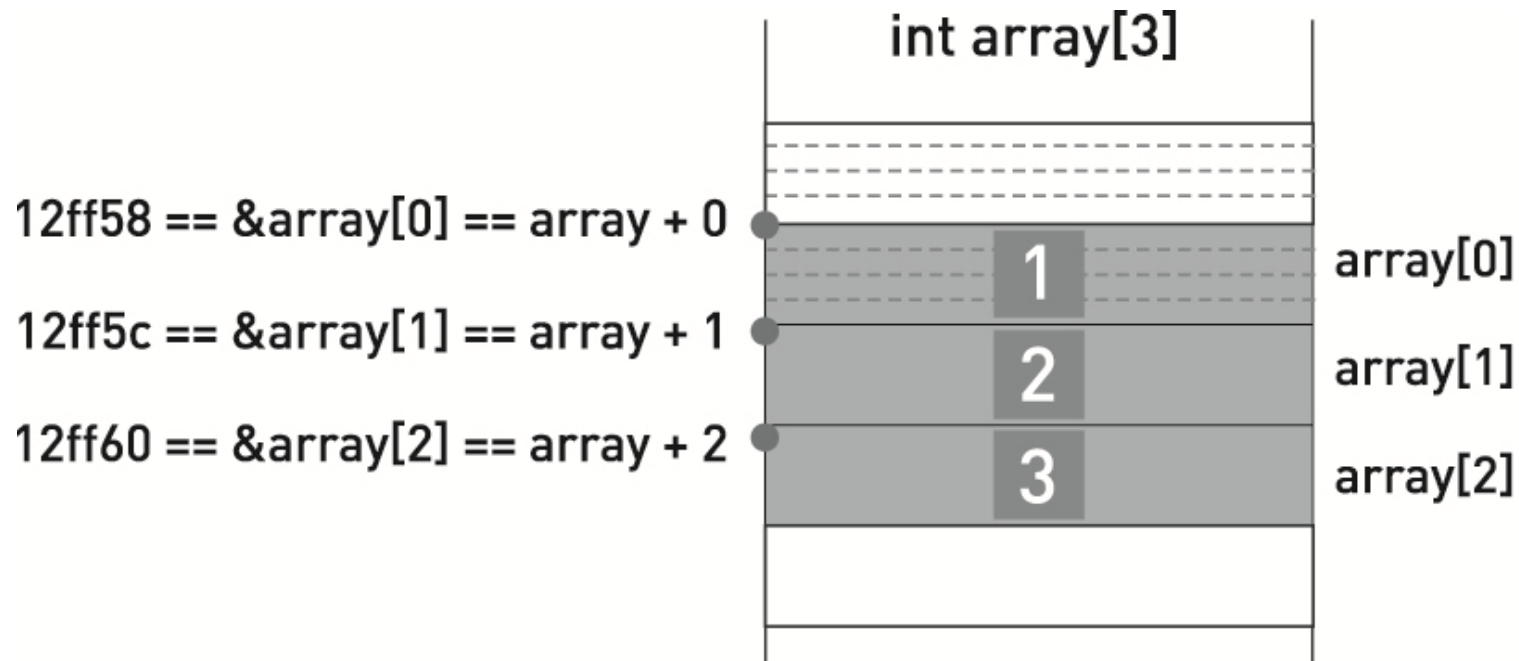
```
#include <stdio.h>
int main(void)
{
    int array[3] = {1,2,3};

    printf("%x %x %x \n", array+0, array+1, array+2);
    printf("%x %x %x \n", &array[0], &array[1], &array[2]);

    return 0;
}
```



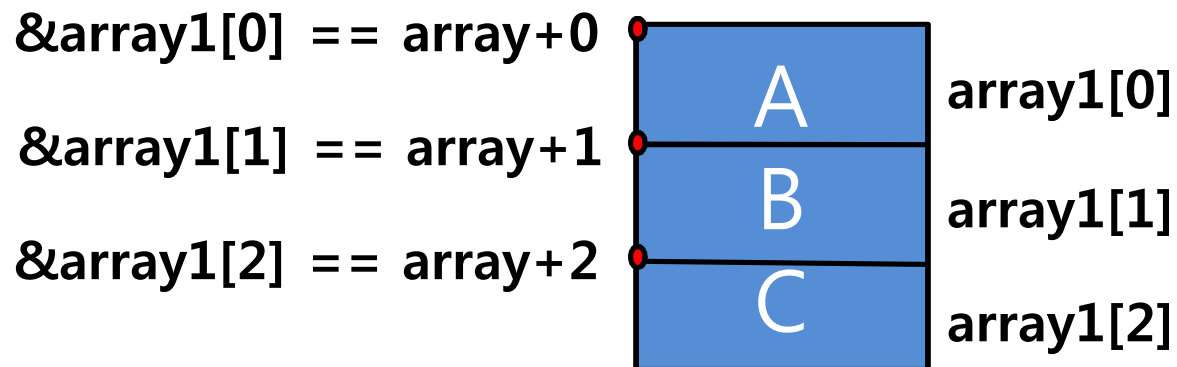
1.2 1차원 배열의 주소와 값의 참조 (6/12)---[1-6.c 분석]



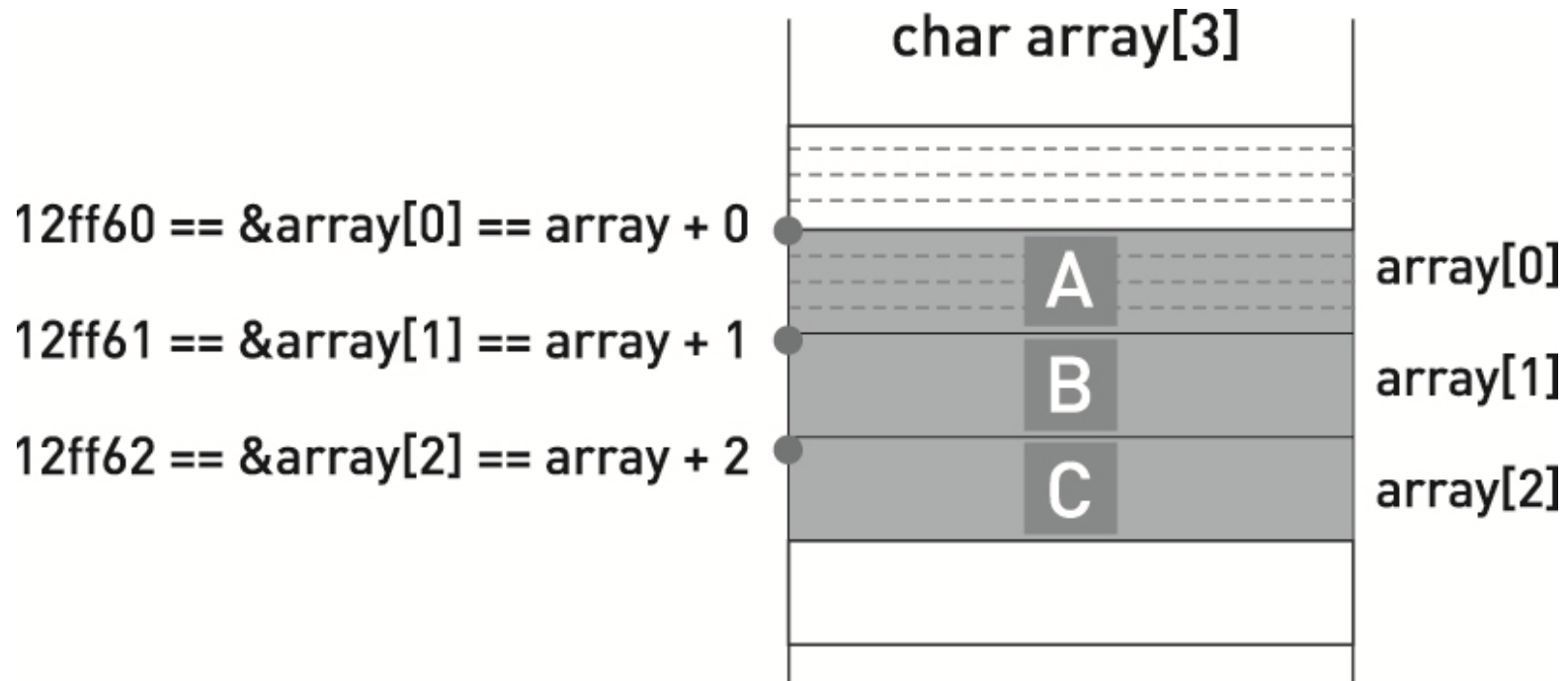
총 12바이트 크기의 연속된 메모리 공간을 할당하며
배열 요소는 0부터 시작

1.2 1차원 배열의 주소와 값의 참조 (7/12)---[1-7.c 실습]

```
#include <stdio.h>
int main(void)
{
    char array[3] = {'A','B','C'};
    printf("%x %x %x \n", array+0, array+1, array+2);
    return 0;
}
```



1.2 1차원 배열의 주소와 값의 참조 (8/12)---[1-7.c 분석]



총 3바이트 크기의 연속된 메모리 공간을 할당하며
배열 요소는 0부터 시작

1.2 1차원 배열의 주소와 값의 참조 (9/12)

▶ * 연산자

✓ 메모리의 주소 앞에 사용된 경우

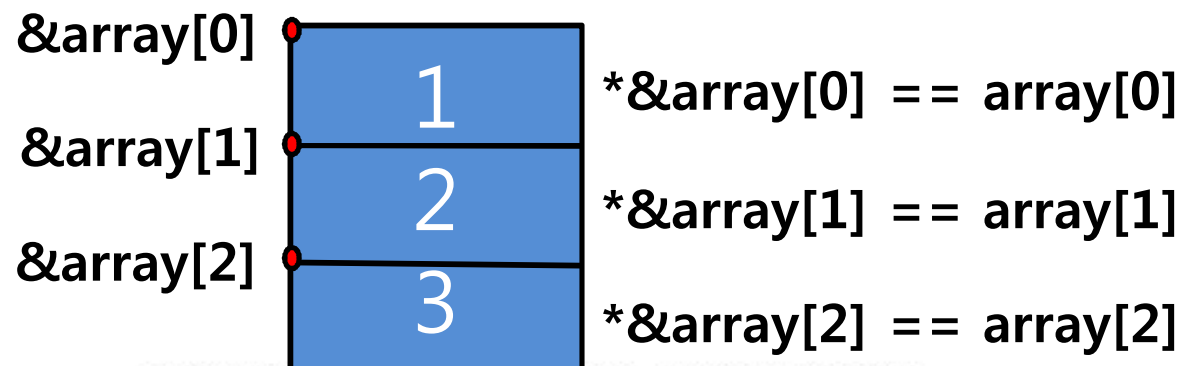
- ' *'는 메모리 공간에 저장된 **값을 참조하는 연산자**이다.

변수의 값 참조	배열 요소의 값 참조
사용법: * &변수이름	사용법: * &배열 요소
<pre>int a=10, b=20; printf("%d \n", *&a); // a의 값 printf("%d \n", *&b); // b의 값</pre>	<pre>int array[2] = {10, 20}; printf("%d \n", *&array[0]); // array[0]의 값 printf("%d \n", *&array[1]); // array[1]의 값</pre>

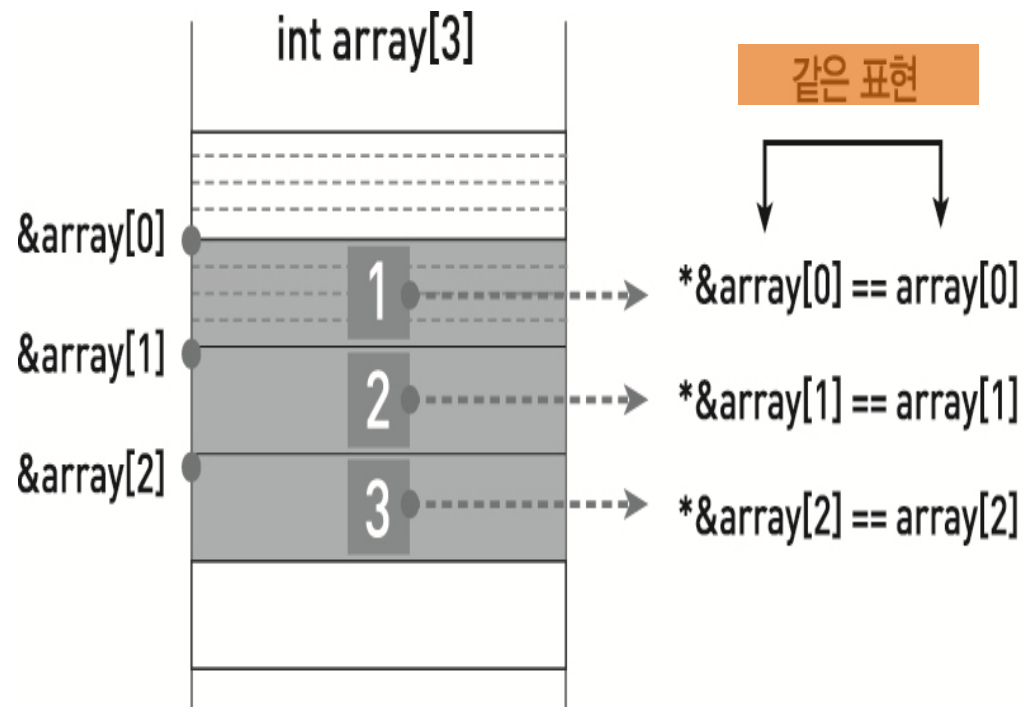
1.2 1차원 배열의 주소와 값의 참조 (10/12)---[1-8.c 실습]

```
#include <stdio.h>
int main(void)
{
    int array[3] = {1,2,3};

    printf("%x %x %x \n", &array[0], &array[1], &array[2]);
    printf("%d %d %d \n", *&array[0], *&array[1], *&array[2]);
    printf("%d %d %d \n", array[0], array[1], array[2]);
    printf("%d %d %d \n", **&array[0], **&array[1], **&array[2]);
    return 0;
}
```



1.2 1차원 배열의 주소와 값의 참조 (11/12)---[1-8.c 분석]



서로 상쇄
↓
`*&array[0] == array[0]`

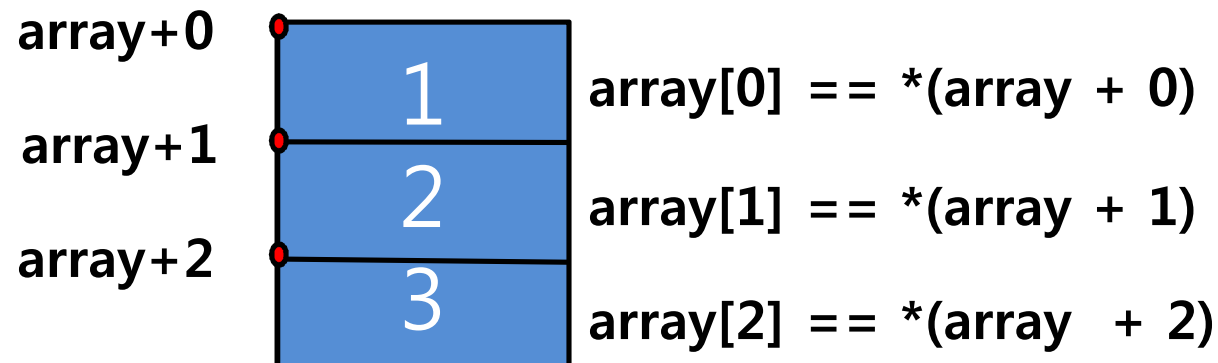
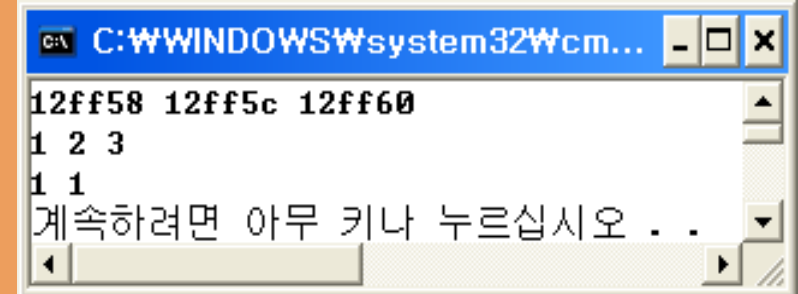
총 12바이트 크기의 연속된 메모리 공간을 할당하며
배열 요소는 0부터 시작

1.2 1차원 배열의 주소와 값의 참조 (12/12)---[1-9.c 실습]

```
#include <stdio.h>
int main(void)
{
    int array[3] = {1,2,3};

    printf("%x %x %x \n", array+0, array+1, array+2);
    printf("%d %d %d \n", *(array+0), *(array+1), *(array+2));
    printf("%d %d \n", *(array+0), *array);

    return 0;
}
```



결론: 반드시 숙지해야 할 사항(1)

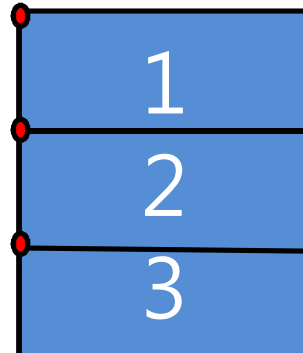
서로 상쇄



`*(array+i) == array[i] == *&array[i]`

```
int array[3]={1,2,3};
```

`array+0`



`array[0] == *(array + 0)`

`array+1`

`array[1] == *(array + 1)`

`array+2`

`array[2] == *(array + 2)`

결론: 반드시 숙지해야 할 사항(2)

서로 상쇄



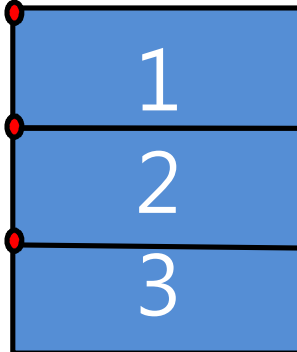
`*(array+i) == array[i] == *&array[i]`

```
int array[3]={1,2,3};
```

`&array[0]`

`&array[1]`

`&array[2]`



`*&array[0] == array[0]`

`*&array[1] == array[1]`

`*&array[2] == array[2]`

결론: 반드시 숙지해야 할 사항(3)

서로 상쇄



`*(array+i) == array[i] == *&array[i]`



```
*array == *(array+0) == array[0]
*(array+1) == array[1]
*(array+2) == array[2]
```

공부한 내용 떠올리기

- ▶ 배열의 정의와 필요성
- ▶ 배열의 선언과 구성 요소
- ▶ 배열에 데이터를 저장하는 방법
- ▶ 배열을 선언할 때 주의할 사항
- ▶ 배열 이름은 배열의 시작 주소
- ▶ 주소와 값을 참조하는 연산자
 - ✓ & 연산자
 - ✓ * 연산자

서로 상쇄
↓
`*(array+i) == array[i] == *&array[i]`

감사의 깨달음 (출처: 사랑과 지혜의 탈무드)

