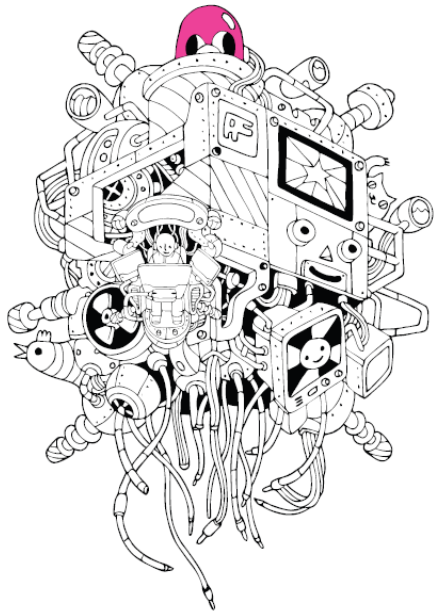
C 프로그래밍

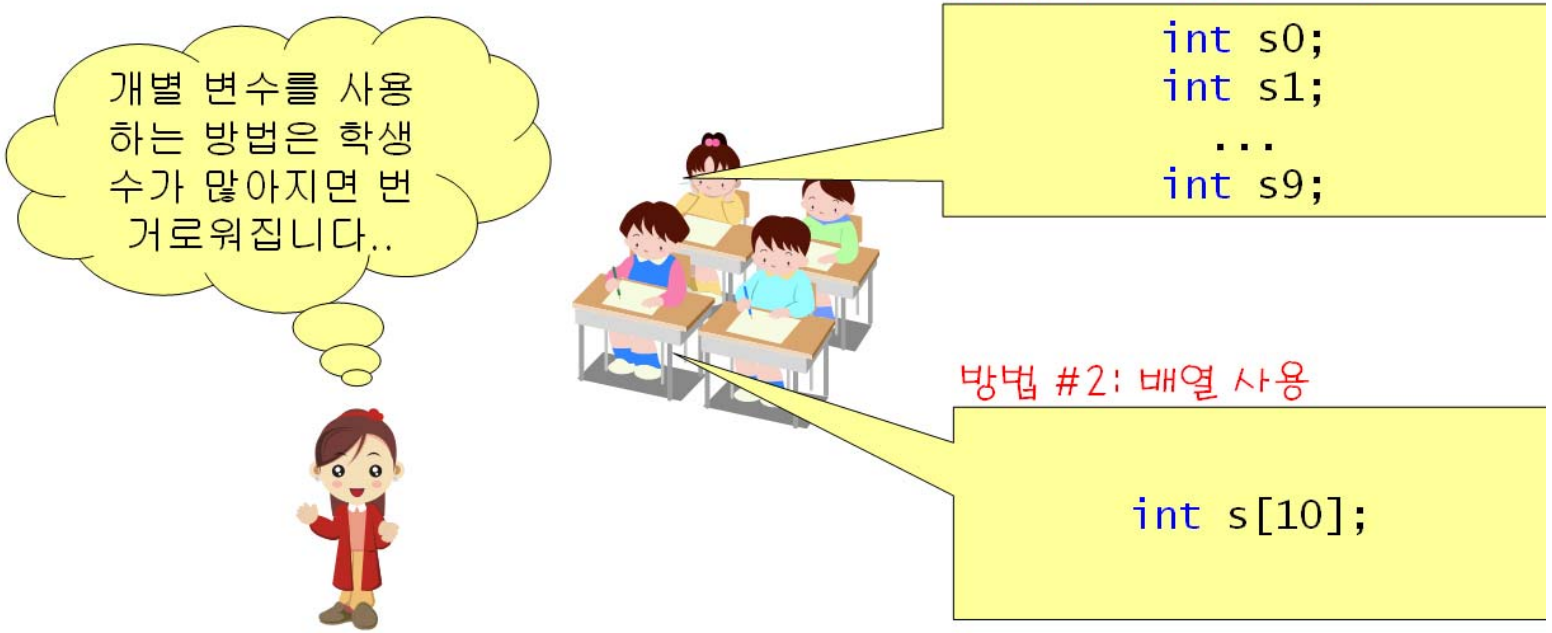


Chapter 11. 1차원 배열

배열의 이해와 배열의 선언 및 초기화 방법

배열의 필요성

- 학생이 10명이 있고 이들의 평균 성적을 계산한다고 가정하자.

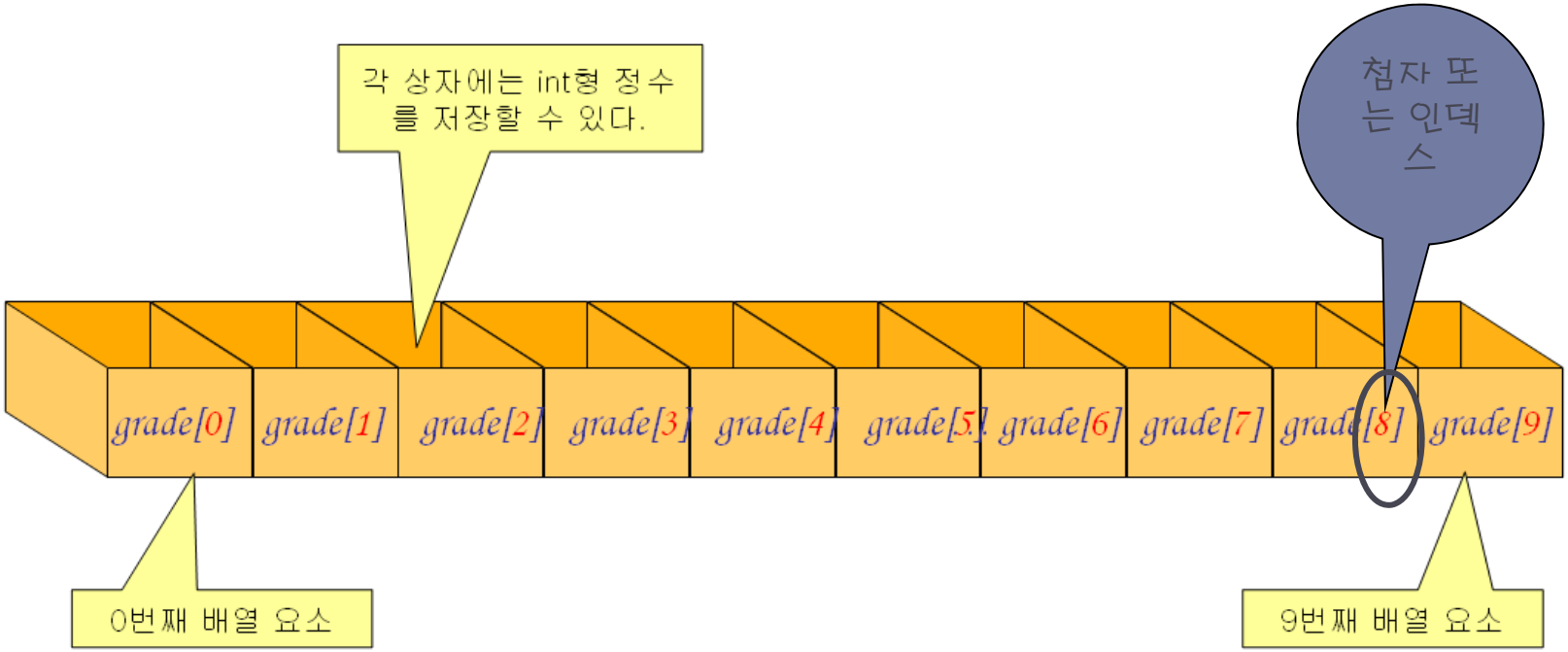


배열이란?

- 배열(array): 동일한 타입의 데이터가 여러 개 저장되어 있는 데이터 저장 장소
- 배열 안에 들어있는 각각의 데이터들은 정수로 되어 있는 번호(첨자)에 의하여 접근
- 배열을 이용하면 여러 개의 값을 하나의 이름으로 처리할 수 있다.

배열 원소와 인덱스

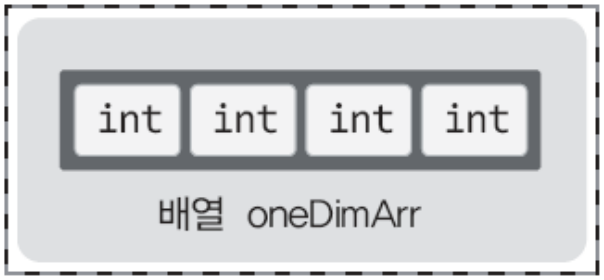
- *인덱스(index):* 배열 원소의 번호



1차원 배열 선언에 필요한 것 세 가지

```
int oneDimArr[4];
```

1차원 배열 선언의 예



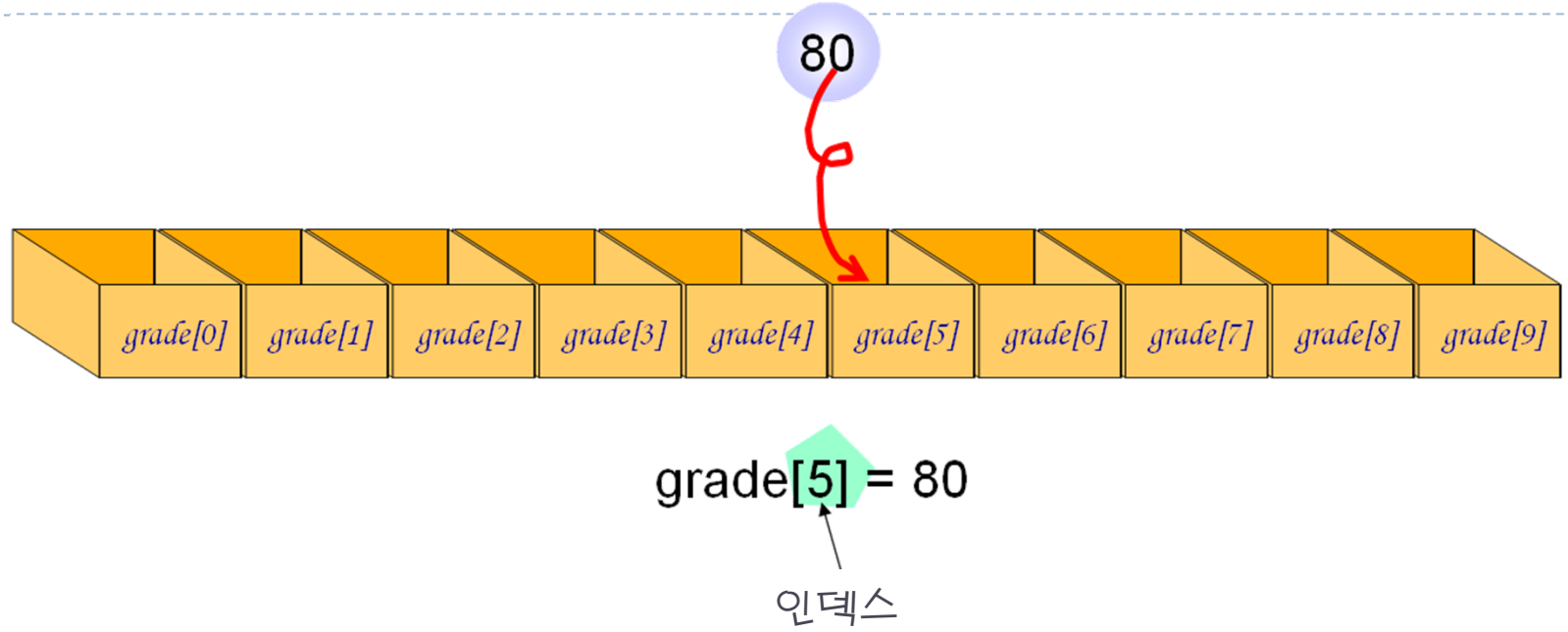
생성되는 배열의 형태

int	배열을 이루는 요소(변수)의 자료형
oneDimArr	배열의 이름
[4]	배열의 길이

인덱스(배열의 번호)는 항상 0부터 시작!!!

```
int  score[60];          // 60개의 int형 값을 가지는 배열 grade
float  cost[12];         // 12개의 float형 값을 가지는 배열 cost
char  name[50];          // 50개의 char형 값을 가지는 배열 name
char  src[10], dst[10];  // 2개의 문자형 배열을 동시에 선언
int   index, days[7];    // 일반 변수와 배열을 동시에 선언
```

배열 원소 접근




```
grade[5] = 80;
grade[1] =  grade[0];
grade[i] = 100;          // i는 정수 변수
grade[i+2] = 100;        // 수식이 인덱스가 된다.
grade[index[3]] = 100;   // index[]는 정수 배열
```

배열 선언 예제

```
#include <stdio.h>

int main(void)
{
    int i;
    int grade[5];
    grade[0] = 10;
    grade[1] = 20;
    grade[2] = 30;
    grade[3] = 40;
    grade[4] = 50;

    for(i=0;i < 5; i++)
        printf("grade[%d]=%d\n",i, grade[i]);
    return 0;
}
```

```
grade[0]=10
grade[1]=20
grade[2]=30
grade[3]=40
grade[4]=50
```

선언된 1차원 배열의 접근

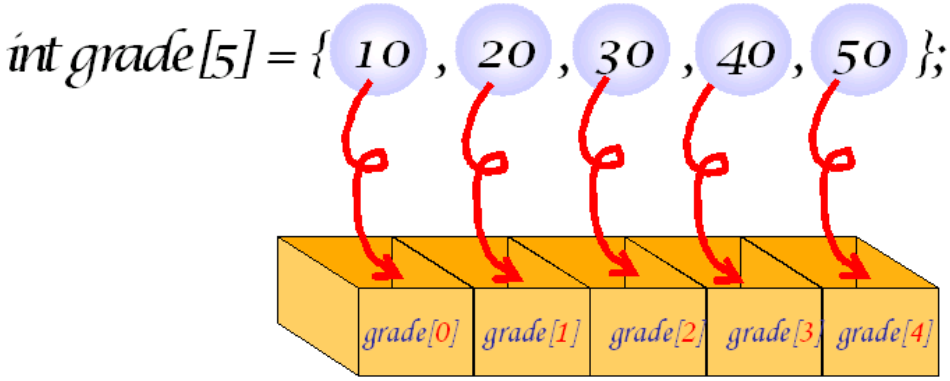
```
int main(void)
{
    int arr[5];
    int sum=0, i;

    arr[0]=10, arr[1]=20, arr[2]=30, arr[3]=40, arr[4]=50;

    for(i=0; i<5; i++)
        sum += arr[i];

    printf("배열요소에 저장된 값의 합: %d \n", sum);
    return 0;
}
```

실행결과

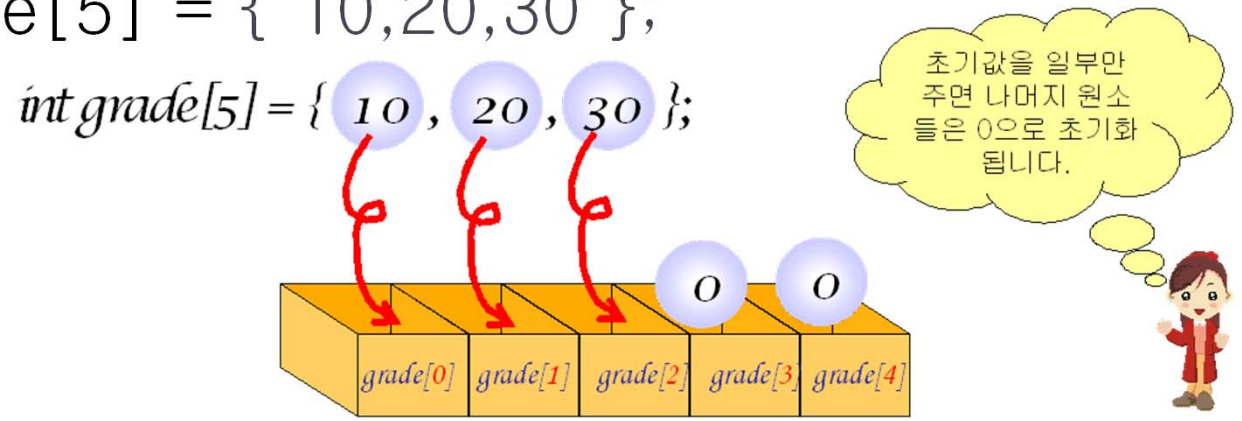
```
배열요소에 저장된 값의 합: 150
```

배열의 초기화 : 선언과 동시에 초기화

- int grade[5] = { 10,20,30,40,50 };



- int grade[5] = { 10,20,30 };



배열! 선언과 동시에 초기화하기

```
int arr2[ ]={1, 2, 3, 4, 5, 6, 7};
```

초기화 리스트는 존재하고 배열의 길이정보 생략된 경우

컴파일러가 배열의 길이정보 채움

```
int arr2[7]={1, 2, 3, 4, 5, 6, 7};
```

배열 선언 예제

```
#include <stdio.h>
int main(void)
{
        int grade[10];
        int i;

        for(i = 0; i < 10; i++)
                grade[i] = 0;

        printf("====================\n");
        printf("인덱스      값\n");
        printf("====================\n");

        for(i = 0; i < 10; i++)
                printf("%5d    %5d\n", i, grade[i]);

        return 0;
}
```

```
====================
인덱스      값
====================
    0      0
    1      0
    2      0
    3      0
    4      0
    5      0
    6      0
    7      0
    8      0
    9      0
```

배열 초기화 예제

```c
#include <stdio.h>
int main(void)
{
	int grade[10] = { 31, 63, 62, 87, 14, 25, 92, 70, 75, 53 };
	int i;

	printf("====================\n");
	printf("인덱스	값\n");
	printf("====================\n");

	for(i = 0; i < 10; i++)
		printf("%5d	%5d\n", i, grade[i]);

	return 0;
}
```

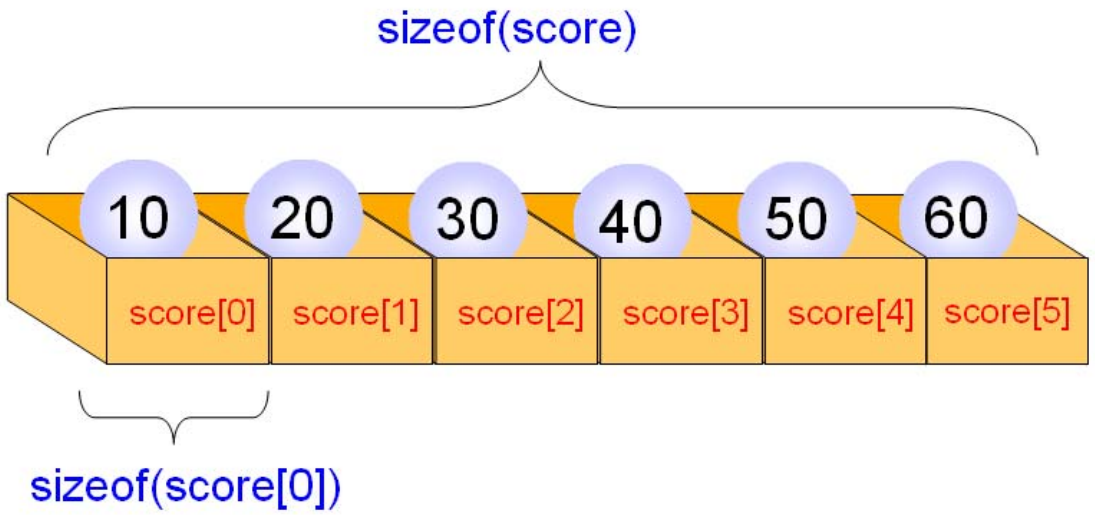
```
====================
인덱스	값
====================
    0	   31
    1	   63
    2	   62
    3	   87
    4	   14
    5	   25
    6	   92
    7	   70
    8	   75
    9	   53
```

배열 원소 참조 예제

```c
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

int main(void)
{
	int grade[SIZE];
	int i;

	for(i = 0; i < SIZE; i++)
		grade[i] = rand() % 100;

	printf("====================\n");
	printf("인덱스	값\n");
	printf("====================\n");

	for(i = 0; i < SIZE; i++)
		printf("%5d	%5d\n", i, grade[i]);
	return 0;
}
```

```
====================
인덱스	값
====================
    0	   41
    1	   67
    2	   34
    3	    0
    4	   69
    5	   24
    6	   78
    7	   58
    8	   62
    9	   64
```

배열 원소의 개수 계산




```
int grade[] = { 1, 2, 3, 4, 5, 6 };
int i, size;

size = sizeof(grade) / sizeof(grade[0]);

for(i = 0; i < size ; i++)
        printf("%d ", grade[i]);
```

배열 원소 개수 자동 계산

1차원 배열의 선언, 초기화 및 접근 관련 예제

```
int main(void)
{
    int arr1[5]={1, 2, 3, 4, 5};
    int arr2[ ]={1, 2, 3, 4, 5, 6, 7};
    int arr3[5]={1, 2};
    int ar1Len, ar2Len, ar3Len, i;

    printf("배열 arr1의 크기: %d \n", sizeof(arr1));
    printf("배열 arr2의 크기: %d \n", sizeof(arr2));
    printf("배열 arr3의 크기: %d \n", sizeof(arr3));

    ar1Len = sizeof(arr1) / sizeof(int);   // 배열 arr1의 길이 계산
    ar2Len = sizeof(arr2) / sizeof(int);   // 배열 arr2의 길이 계산
    ar3Len = sizeof(arr3) / sizeof(int);   // 배열 arr3의 길이 계산

    for(i=0; i<ar1Len; i++)
        printf("%d ", arr1[i]);
    printf("\n");

    for(i=0; i<ar2Len; i++)
        printf("%d ", arr2[i]);
    printf("\n");

    for(i=0; i<ar3Len; i++)
        printf("%d ", arr3[i]);
    printf("\n");
    return 0;
}
```

sizeof 연산의 결과로
배열의 바이트 크기정보 반환

배열의 길이를 계산하는 방식에 주목!

배열이기에 for문을 통한 순차적 접근이 가능하다.

다수의 변수라면 반복문을 통한 순차적 접근 불가능!

실행결과

```
배열 arr1의 크기: 20
배열 arr2의 크기: 28
배열 arr3의 크기: 20
1 2 3 4 5
1 2 3 4 5 6 7
1 2 0 0 0
```

잘못된 인덱스로 접근하는 경우

```c
#include <stdio.h>
#define SIZE 5

int main(void)
{
	int array[SIZE] = {1, 2, 3, 4, 5};
	int i;

	for(i = 0; i <= SIZE; i++)
		printf("array[%d]  %d₩n", i, array[i]);

	return 0;
}
```

```
array[0]	1
array[1]	2
array[2]	3
array[3]	4
array[4]	5
array[5]	1245120
```

배열의 복사

```
int grade[SIZE];
int score[SIZE];

score = grade;              // 컴파일 오류!
```

잘못된 방법

```
#include <stdio.h>
#define SIZE 5

int main(void)
{
        int i;
        int a[SIZE] = {1, 2, 3, 4, 5};
        int b[SIZE];

        for(i = 0; i < SIZE; i++)
                b[i] = a[i];

        return 0;
}
```

올바른 방법

배열의 비교

```c
#include <stdio.h>
#define SIZE 5

int main(void)
{
	int i;
	int a[SIZE] = { 1, 2, 3, 4, 5 };
	int b[SIZE] = { 1, 2, 3, 4, 5 };

	if( a == b )		// ① 올바르지 않은 배열 비교
		printf("잘못된 결과입니다.\n");
	else
		printf("잘못된 결과입니다.\n");

	for(i = 0; i < SIZE ; i++) // ② 올바른 배열 비교
	{
		if ( a[i] != b[i] )
		{
			printf("a[]와 b[]는 같지 않습니다.\n");
			return 0;
		}
	}
	printf("a[]와 b[]는 같습니다.\n");
	return 0;
}
```

배열 원소 역순 출력

```c
#include <stdio.h>
#define SIZE 5

int main(void)
{
	int data[SIZE];
	int i;

	for(i = 0; i < SIZE; i++)		// 정수를 입력받는 루프
	{
		printf("정수를 입력하시오:");
		scanf("%d", &data[i]);
	}

	for(i = SIZE - 1;i >= 0; i--)	// 역순으로 출력하는 루프
	{
		printf("%d₩n", data[i]);
	}
	return 0;
}
```

```
정수를 입력하시오:10
정수를 입력하시오:20
정수를 입력하시오:30
정수를 입력하시오:40
정수를 입력하시오:50
50
40
30
20
10
```

예제

```c
#include <stdio.h>
#define STUDENTS 5

int main(void)
{
	int grade[STUDENTS] = { 30, 20, 10, 40, 50 };
	int i, s;

	for(i = 0; i < STUDENTS; i++)
	{
		printf("번호 %d: ", i);
		for(s = 0; s < grade[i]; s++)
			printf("*");
		printf("₩n");
	}

	return 0;
}
```

```
번호 0: ******************************
번호 1: ********************
번호 2: **********
번호 3: ****************************************
번호 4: **************************************************
```

최소값 탐색

```c
#include <stdio.h>
#define SIZE 10

int main(void)
{
	int grade[SIZE];
	int i, min;

	for(i = 0; i < SIZE; i++)
	{
		printf("성적을 입력하시오: ");
		scanf("%d", &grade[i]);
	}

	min = grade[0];

	for(i = 1; i < SIZE; i++)
	{
		if( grade[i] < min )
			min = grade[i];
	}

	printf("최소값은 %d입니다.₩n", min);
	return 0;
}
```

```
숫자를 입력하시오: 50
숫자를 입력하시오: 40
숫자를 입력하시오: 30
숫자를 입력하시오: 20
숫자를 입력하시오: 10
숫자를 입력하시오: 20
숫자를 입력하시오: 30
숫자를 입력하시오: 40
숫자를 입력하시오: 60
숫자를 입력하시오: 70
최소값은 10입니다.
```

빈도 계산

```c
#include <stdio.h>
#define SIZE 101

int main(void)
{
        int freq[SIZE];
        int i, score;

        for(i = 0; i < SIZE; i++)
                freq[i] = 0;

        while(1)
        {
                printf("숫자를 입력하시오(종료-1): ");
                scanf("%d", &score);
                if (score < 0) break;
                freq[score]++;
        }

        printf("값   빈도\n");

        for(i = 0; i < SIZE; i++)
                printf("%3d     %3d \n", i, freq[i]);

        return 0;
}
```

```
숫자를 입력하시오(종료 -1): 0
숫자를 입력하시오(종료 -1): 1
숫자를 입력하시오(종료 -1): 99
숫자를 입력하시오(종료 -1): 100
숫자를 입력하시오(종료 -1): 100
숫자를 입력하시오(종료 -1): -1
값    빈도
  0     1
  1     1
  2     0
 ...
 98     0
 99     1
100     2
```

주사위면 빈도 계산

```c
#include <stdio.h>
#include <stdlib.h>

#define SIZE 6

int main(void)
{
        int freq[SIZE] = { 0 };              // 주사위의 면의 빈도를 0으로 한다.
        int i;

        for(i = 0; i < 10000; i++)          // 주사위를 10000번 던진다.
                ++freq[ rand() % 6 ];        // 해당면의 빈도를 하나 증가한다.

        printf("====================\n");
        printf("면      빈도\n");
        printf("====================\n");

        for(i = 0; i < SIZE; i++)
                printf("%3d     %3d \n", i, freq[i]);

        return 0;
}
```

```
====================
면    빈도
====================
  0    1657
  1    1679
  2    1656
  3    1694
  4    1652
  5    1662
```

배열과 함수

```c
#include <stdio.h>
#define STUDENTS 5
int get_average(int score[], int n);     // ①

int main(void)
{
        int grade[STUDENTS] = { 1, 2, 3, 4, 5 };
        int avg;

        avg = get_average(grade, STUDENTS);
        printf("평균은 %d입니다.\n", avg);
        return 0;
}

int get_average(int score[], int n)      // ②
{
        int i;
        int sum = 0;

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

배열이 인수인 경우, 참조에 의한 호출

배열의 원본이 score[]로 전달

배열이 함수의 인수인 경우 1/2

```c
#include <stdio.h>
#define SIZE 7

void square_array(int a[], int size);
void print_array(int a[], int size);
void square_element(int e);

int main(void)
{
	int list[SIZE] = { 1, 2, 3, 4, 5, 6, 7 } ;

	print_array(list, SIZE);
	square_array(list, SIZE);  // 배열은 원본이 전달된다.
	print_array(list, SIZE);

	printf("%3d\n", list[6]);
	square_element(list[6]);   // 배열 요소는 복사본이 전달된다.
	printf("%3d\n", list[6]);

	return 0;
}
```

배열이 함수의 인수인 경우 2/2

```
void square_array(int a[], int size)
{
        int i;

        for(i = 0; i < size; i++)
                a[i] = a[i] * a[i];
}
void square_element(int e)
{
        e = e * e;
}
void print_array(int a[], int size)
{
        int i;

        for(i = 0; i < size; i++)
                printf("%3d ", a[i]);
        printf("\n");
}
```

배열 원소의 사본이 e로 전달

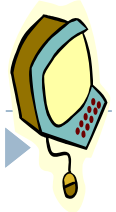
배열의 원본이 a[]로 전달

```
  1   2   3   4   5   6   7
  1   4   9  16  25  36  49
  7
  7
```

원본 배열의 변경을 금지하는 방법

```
#include <stdio.h>
#define SIZE 20

void copy_array(char dest[], const char src[], int count);

int main(void)
{
	char s[SIZE] = { 'H', 'E', 'L', 'L', 'O', '\0' };
	char d[SIZE];

	copy_array(d, s, SIZE);
	printf("%s\n", d);
	return 0;
}
void copy_array(char dest[], const char src[], int size)
{
	int i;
	for(i = 0; i < size; i++)
	{
		dest[i] = src[i];
	}
}
```

배열 원본의 변경 금지



```
HELLO
```

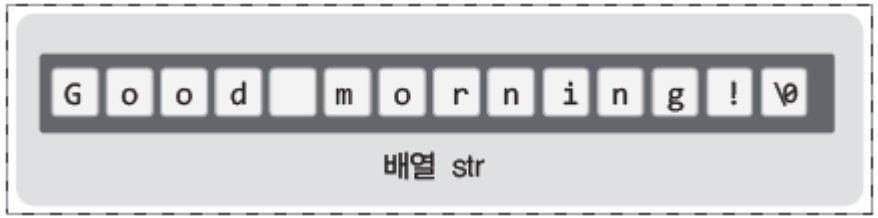
배열을 이용한 문자열 변수의 표현

char형 배열의 문자열 저장과 널 문자

```
char str[14]="Good morning!";
```

배열에 문자열 저장

저장결과



문자열의 끝에 널 문자라 불리는 \0가 삽입되었음에 주목! 널 문자는 문자열의 끝을 의미한다.

```
int main(void)
{
    char str[]="Good morning!";
    printf("배열 str의 크기: %d \n", sizeof(str));
    printf("널 문자 문자형 출력: %c \n", str[13]);
    printf("널 문자 정수형 출력: %d \n", str[13]);

    str[12]='?';    // 배열 str에 저장된 문자열 데이터는 변경 가능!
    printf("문자열 출력: %s \n", str);
    return 0;
}
```

실행결과

```
배열 str의 크기: 14
널 문자 문자형 출력:
널 문자 정수형 출력: 0
문자열 출력: Good morning?
```

널 문자와 공백 문자의 비교

```
int main(void)
{
     char nu = '\0';    // 널 문자 저장
     char sp = ' ';     // 공백 문자 저장
     printf("%d %d", nu, sp);    // 0과 32 출력
     return 0;
}
```

널 문자를 %c를 이용해서 출력 시 아무것도 출력되지 않는다. 그렇다고 해서 널 문자가 공백 문자는 아니다.

널 문자의 아스키 코드 값은 0이고, 공백 문자의 아스키 코드 값은 32이다.

널 문자는 모니터 출력에서 의미를 갖지 않는다. 그래서 아무것도 출력이 되지 않을 뿐이다.

scanf 함수를 이용한 문자열의 입력

```
int main(void)
{
    char str[50];
    int idx=0;

    printf("문자열 입력: ");
    scanf("%s", str);  // 문자열을 입력 받아서 배열 str에 저장!
    printf("입력 받은 문자열: %s \n", str);

    printf("문자 단위 출력: ");
    while(str[idx] != '\0')
    {
        printf("%c", str[idx]);
        idx++;
    }
    printf("\n");
    return 0;
}
```

scanf 함수의 호출을 통해서 입력 받은 문자열의 끝에도 널 문자가 존재함을 확인하기 위한 문장

scanf 함수를 이용해서 문자열 입력 시 서식문자 %s를 사용한다.

scanf("%s", str);

위와 같이 배열이름 str의 앞에는 & 연산자를 붙이지 않는다.

실행결과

```
문자열 입력: Simple
입력 받은 문자열: Simple
문자 단위 출력: Simple
```

```
char arr1[ ] = {'H', 'i', '~'};
char arr2[ ] = {'H', 'i', '~', '\0'};
```

arr1은 문자열이 아닌 문자 배열, 반면 arr2는 문자열!
널 문자의 존재여부는 문자열의 판단여부가 된다.

문자열의 끝에 널 문자가 필요한 이유

문자열의 시작은 판단할 수 있어도 문자열의 끝은 판단이 불가능하다! 때문에 문자열의 끝을 판단할 수 있도록 널 문자가 삽입이 된다.

```
int main(void)
{
    char str[50]="I like C programming";
    printf("string: %s \n", str);

    str[8]='\0';    // 9번째 요소에 널 문자 저장
    printf("string: %s \n", str);

    str[6]='\0';    // 7번째 요소에 널 문자 저장
    printf("string: %s \n", str);

    str[1]='\0';    // 2번째 요소에 널 문자 저장
    printf("string: %s \n", str);
    return 0;
}
```

배열의 시작위치에 문자열이 저장되기 시작한다. 따라서 시작위치는 확인이 가능하다. 하지만 배열의 끝이 문자열의 끝은 아니므로 널 문자가 삽입되지 않으면 문자열의 끝은 확인이 불가능하다.

실행결과

```
string: I like C programming
string: I like C
string: I like
string: I
```

위 예제에서 보이듯이 printf 함수도 배열 str의 시작위치를 기준으로해서 널 문자를 만날 때까지 출력을 진행한다. 따라서 널 문자가 없으면 printf 함수도 문자열의 끝을 알지 못한다.

scanf 함수의 문자열 입력 특성

```
int main(void)
{
    char str[50];
    int idx=0;

    printf("문자열 입력: ");
    scanf("%s", str);  // 문자열을 입력 받아서 배열 str에 저장!
    printf("입력 받은 문자열: %s \n", str);

    printf("문자 단위 출력: ");
    while(str[idx] != '\0')
    {
        printf("%c", str[idx]);
        idx++;
    }
    printf("\n");
    return 0;
}
```

앞서 보인 왼편의 예제를 실행할 때 다음과 같이 문자열을 입력하면

He is my friend

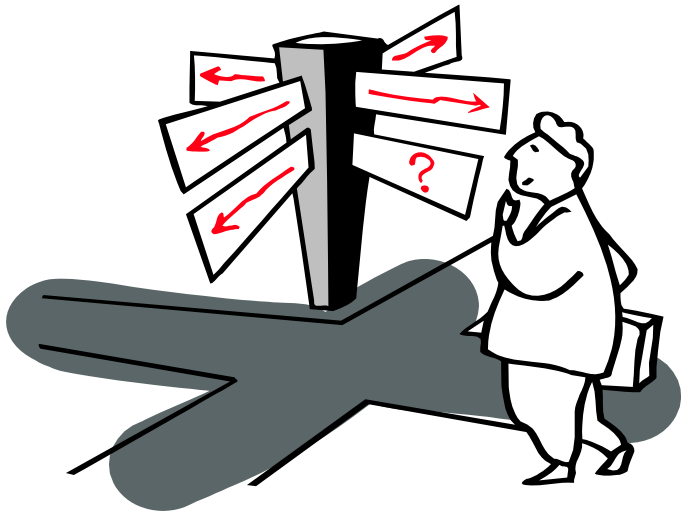
다음의 실행결과를 보인다.

입력 받은 문자열: He

문자 단위 출력: He

scanf 함수는 공백을 기준으로 데이터의 수를 구분한다. 따라서 공백을 포함하는 문자열을 한번의 scanf 함수호출을 통해서 읽어 들이지는 못한다.

공백을 포함하는 문자열의 입력에 사용되는 함수는 이후에 별도로 설명합니다.



Chapter 11이 끝났습니다. 질문 있으신지요?