

CHAPTER

10

배열

- 배열의 필요성과 배열을 선언하고 초기화하는 방법을 학습한다.
- 인덱스 개념과 인덱스를 이용하여 배열의 요소를 참조하는 방법을 학습한다.
- 반복 구조를 이용하여서 배열 원소들에 대하여 동일한 처리를 반복 적용하는 방법을 학습한다.
- 다차원 배열의 개념과 선언방법을 학습한다.
- 정렬과 탐색 알고리즘을 살펴본다.

Contents

3

10.1

배열이란?

10.2

배열의 초기화

10.3

배열과 함수

10.4

정렬

10.5

탐색

10.6

2차원 배열

배열이란?

4

- 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

5

- 배열을 사용하면 한 번에 여러 개의 변수를 생성할 수 있다.
- `int s[10];`



변수

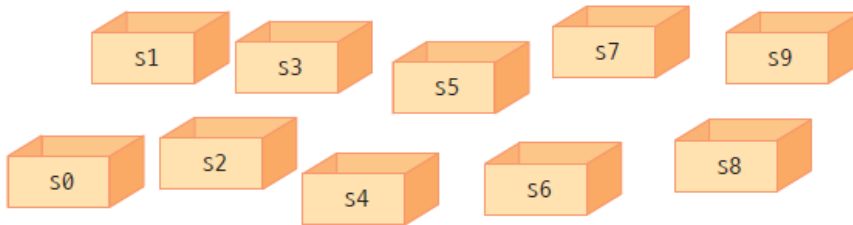


아파트

배열의 필요성

6

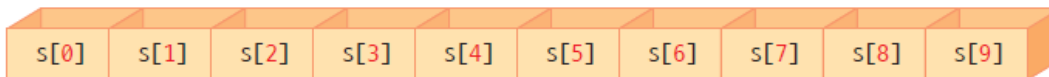
```
// 일반 변수 사용  
int s0;  
int s1;  
...  
int s9;
```



별도의 이름을 가지니
조작하기가 어렵군!

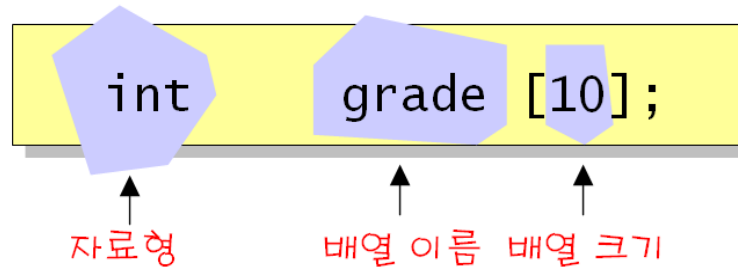
```
// 배열 사용  
int s[10];
```

⇒ 40byte



배열의 선언

7

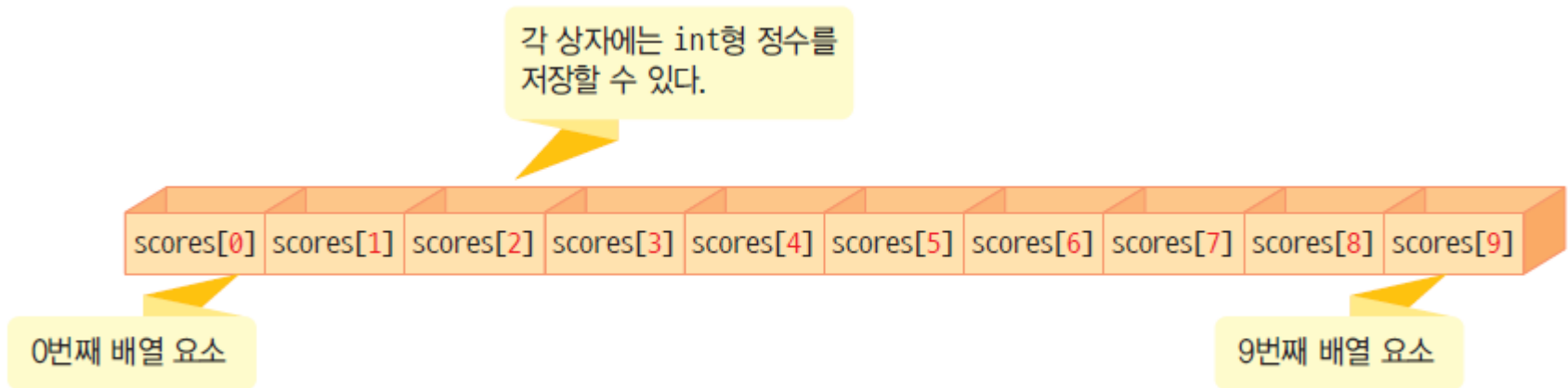


- 자료형: 배열 원소들이 int형라는 것을 의미
- 배열 이름: 배열을 사용할 때 사용하는 이름이 grade
- 배열 크기: 배열 원소의 개수가 10개
- 인덱스(배열 번호)는 항상 0부터 시작한다.

배열 원소와 인덱스

8

- 인덱스(index): 배열 원소의 번호



배열 선언의 예

9

```
int score[60];
```

// 60개의 int형 값을 가지는 배열 score

```
float cost[12];
```

// 12개의 float형 값을 가지는 배열 cost

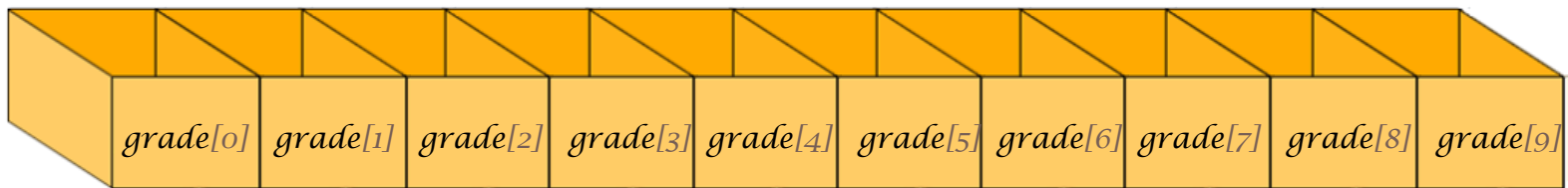
```
char name[50];
```

// 50개의 char형 값을 가지는 배열 name

배열 원소 접근

10

80



`grade[5] = 80`

```
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

11

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

```
int main(void)
```

```
{
```

```
    int i;
```

```
    int score[5];
```

```
    score[0] = 10;
```

```
    score[1] = 20;
```

```
    score[2] = 30;
```

```
    score[3] = 40;
```

```
    score[4] = 50;
```

```
    for(i=0; i < 5; i++)
```

```
        printf("score[%d]=%d\n", i, score[i]);
```

```
    return 0;
```

```
}
```

```
score[0]=10
```

```
score[1]=20
```

```
score[2]=30
```

```
score[3]=40
```

```
score[4]=50
```

배열과 반복문

12

- 배열의 가장 큰 장점은 반복문을 사용하여 배열의 원소를 **간편하게 처리할 수 있다**는 점

```
scores[0] = 0;  
scores[1] = 0;  
scores[2] = 0;  
scores[3] = 0;  
scores[4] = 0;
```

배열 요소의 개수만큼 되풀이 하여야 한다.
동일한 코드가 반복되어서 소스의 크기가 커진다

```
#define SIZE 5  
...  
for(int i=0 ; i<SIZE ; i++)  
    scores[i] = 0;
```

배열의 크기가 변경되면 SIZE의 정의만 변경하면 된다

배열 선언 예제

13

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

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

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

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

    return 0;
}
```

```
scores[0]=41
scores[1]=67
scores[2]=34
scores[3]=0
scores[4]=69
```

배열 선언 예제

14

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

int main(void)
{
    int i;
    int scores[SIZE];
    printf("5명의 점수를 입력하시오\n");

    for(i = 0; i < SIZE; i++)
        scanf("%d", &scores[i]);

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

5명의 점수를 입력하시오

23

35

67

45

21

scores[0]=23

scores[1]=35

scores[2]=67

scores[3]=45

scores[4]=21

배열 선언 예제

15

```
#include <stdio.h>
#define STUDENTS 5
int main(void)
{
    int scores[STUDENTS];
    int sum = 0;
    int i, average;
    for(i = 0; i < STUDENTS; i++)
    {
        printf("학생들의 성적을 입력하시오: ");
        scanf("%d", &scores[i]);
    }
    for(i = 0; i < STUDENTS; i++)
        sum += scores[i];
    average = sum / STUDENTS;
    printf("성적 평균 = %d\n", average);

    return 0;
}
```

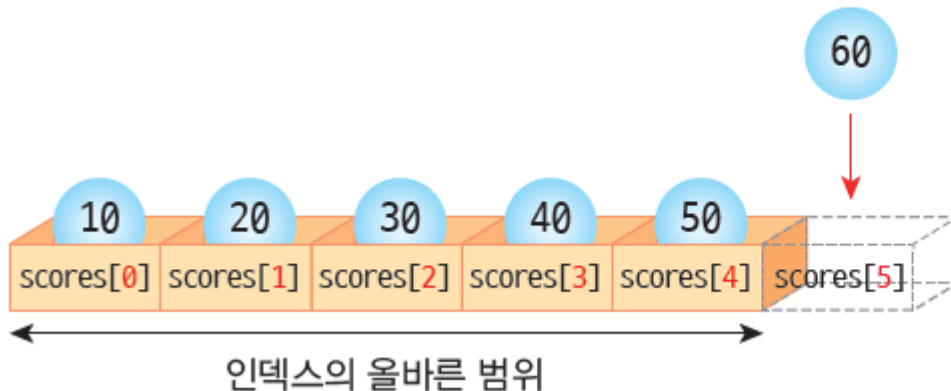
학생들의 성적을 입력하시오: 10
학생들의 성적을 입력하시오: 20
학생들의 성적을 입력하시오: 30
학생들의 성적을 입력하시오: 40
학생들의 성적을 입력하시오: 50
성적 평균 = 30

잘못된 인덱스 문제

16

- 인덱스가 배열의 크기를 벗어나게 되면 프로그램에 ~~치명~~
~~적인~~ 오류를 발생시킨다.
- C에서는 프로그래머가 인덱스가 범위를 벗어나지 않았는지를 확인하고 책임을 져야 한다.

```
int score[5];  
...  
score[5] = 60;    // 치명적인 오류!
```



존재하지 않는 곳에 데이터를 저장하면 안됩니다.



잘못된 인덱스 예제

17

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

    grade[0]=10;
    grade[1]=20;
    grade[2]=30;
    grade[3]=40;
    grade[4]=50;
    grade[5]=60;

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

    return 0;
}
```

시스템에 심각한 오류가 발생할 수도 있다.

Contents

18

10.1

배열이란?

10.2

배열의 초기화

10.3

배열과 함수

10.4

정렬

10.5

탐색

10.6

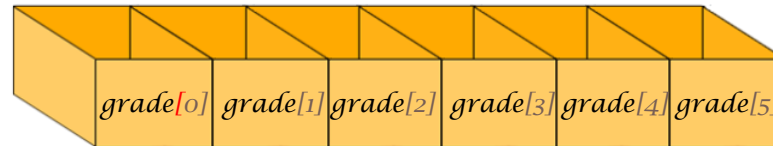
2차원 배열

배열의 초기화

19

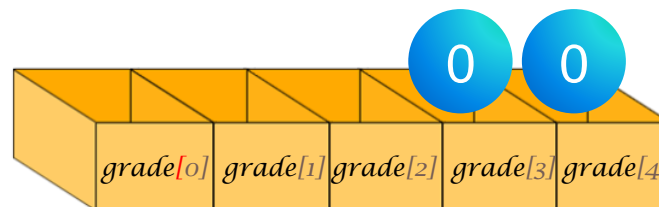
□ `int grade[6] = { 10,20,30,40,50,60 };`

`int grade[5] = { 10, 20, 30, 40, 50, 60 };`



□ `int grade[5] = { 10,20,30 };`

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



초기값을 일부만 주면 나머지 원소들은 0으로 초기화됩니다.



배열의 초기화

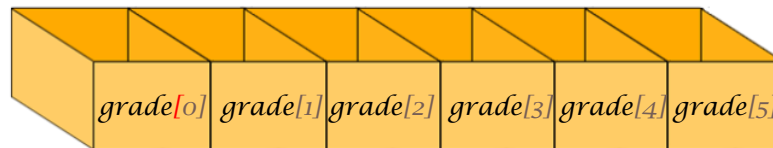
20

- 배열의 크기가 주어지지 않으면 자동으로 초기값의 개수만큼의 배열의 크기로 잡힌다.

6

┌───────────┐

int grade[] = { 10, 20, 30, 40, 50, 60 };



배열 초기화 예제

21

```
#include <stdio.h>
int main(void)
{
    int scores[5] = { 31, 63, 62, 87, 14 };
    int i;

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

    return 0;
}
```

```
scores[0] = 31
scores[1] = 63
scores[2] = 62
scores[3] = 87
scores[4] = 14
```

배열 초기화 예제

22

```
#include <stdio.h>
int main(void)
{
    int scores[5] = { 31, 63 };
    int i;

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

    return 0;
}
```

```
scores[0] = 31
scores[1] = 63
scores[2] = 0
scores[3] = 0
scores[4] = 0
```

배열 초기화 예제

23

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

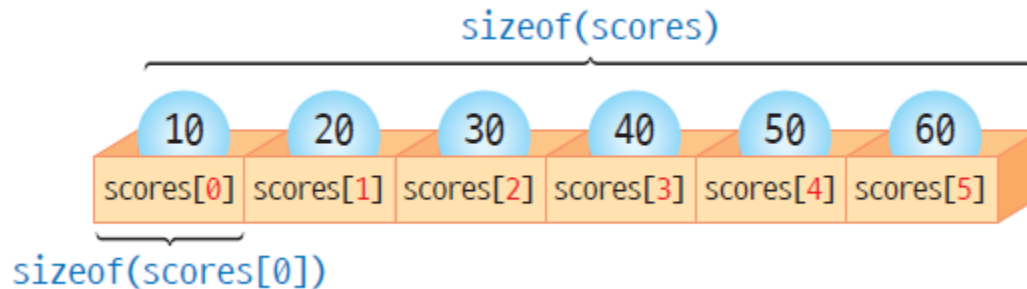
    for(i = 0; i < 5; i++)
        printf("scores[%d] = %d\n", i, scores[i]);

    return 0;
}
```

```
scores[0]=4206620
scores[1]=0
scores[2]=4206636
scores[3]=2018779649
scores[4]=1
```

배열 요소의 개수 계산

24



```
int scores[] = { 10, 20, 30, 40, 50, 60 };
```

```
int i, size;
```

```
size = sizeof(scores) / sizeof(scores[0]);
```

배열 원소 개수 자동 계산

```
for(i = 0; i < size ; i++)
```

```
    printf("%d ", scores[i]);
```


배열의 복사

25

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

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

잘못된 방법

```
score = grade;
```

// 컴파일 오류!

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

```
#define SIZE 5
```

```
int main(void)
```

```
{
```

```
    int i;
```

```
    int a[SIZE] = {1, 2, 3, 4, 5};
```

```
    int b[SIZE];
```

올바른 방법

```
    for(i = 0; i < SIZE; i++)
```

```
        b[i] = a[i];
```

```
    return 0;
```

```
}
```

원소를 일일이
복사한다

배열의 비교

26

```
#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 올바른지 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;
```

```
}
```



원소를 하나씩
비교한다

Lab: 주사위 던지기

27

- 이 실습에서는 주사위를 1000번 던져서 각 면이 나오는 횟수를 출력하여 보자.



주사위 던지기

28

```
#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

Lab: 극장 예약 시스템

29

- 배열을 이용하여 간단한 극장 예약 시스템을 작성
- 좌석은 10개
- 먼저 좌석 배치표를 보여준다.
- 예약이 끝난 좌석은 1로, 예약이 안 된 좌석은 0으로 나타낸다.



실행 결과

30

좌석을 예약하시겠습니까?(y 또는 n) y

1 2 3 4 5 6 7 8 9 10

0 0 0 0 0 0 0 0 0 0

몇번째 좌석을 예약하시겠습니까?1

예약되었습니다.

좌석을 예약하시겠습니까?(y 또는 n) y

1 2 3 4 5 6 7 8 9 10

1 0 0 0 0 0 0 0 0 0

몇번째 좌석을 예약하시겠습니까?1

이미 예약된 자리입니다. 다른 좌석을 선택하세요

좌석을 예약하시겠습니까?(y 또는 n) n

알고리즘

31

while(1)

 사용자로부터 예약 여부(y 또는 n)를 입력받는다.

 if 입력 == 'y'

 현재의 좌석 배치표 seats[]를 출력한다.

 좌석 번호 i를 사용자로부터 입력받는다.

 if 좌석번호가 올바르면

 seats[i]=1

 else

 에러 메시지를 출력한다.

 else

 종료한다.

극장 예약 시스템

32

```
#include <stdio.h>
#define SIZE 10
int main(void)
{
    char ans1;
    int ans2, i;
    int seats[SIZE] = {0};
    while(1)
    {
        printf("좌석을 예약하시겠습니까?(y 또는n) ");
        scanf(" %c",&ans1);
        if(ans1 == 'y')
        {
            printf("-----\n");
            printf(" 1 2 3 4 5 6 7 8 9 10\n");
            printf("-----\n");
            for(i = 0; i < SIZE; i++)
                printf(" %d", seats[i]);
            printf("\n");
            printf("몇번째 좌석을 예약하시겠습니까);
            scanf("%d",&ans2);
```

현재 좌석 예약
상태 출력



극장 예약 시스템

33

```
        if(ans2 <= 0 || ans2 > SIZE) {
            printf("1부터 10사이의 숫자를 입력하세요\n");
            continue;
        }
        if(seats[ans2-1] == 0) {    // 예약되지 않았으면
            seats[ans2-1] = 1;
            printf("예약되었습니다.\n");
        }
        else                        // 이미 예약되었으면
            printf("이미 예약된 자리입니다.\n");
    }
    else if(ans1 == 'n')
        return 0;
}
return 0;
}
```

도전문제

34

- 위의 프로그램에서는 한명만 예약할 수 있다. 하지만 극장에 혼자서 가는 경우는 드물다. 따라서 한번에 2명을 예약할 수 있도록 위의 프로그램을 변경하여 보자.



Lab: 최소값 찾기

35

- 우리는 인터넷에서 상품을 살 때, 가격 비교 사이트를 통하여 가장 싼 곳을 검색한다.
- 일반적으로 배열에 들어 있는 정수 중에서 최소값을 찾는 문제와 같다.



실행 결과

36



1 2 3 4 5 6 7 8 9 10

28 81 60 83 67 10 66 97 37 94

최소값은 10입니다.

알고리즘

37

배열 prices[]의 원소를 난수로 초기화한다.

일단 첫 번째 원소를 최소값 minium이라고 가정한다.

```
for(i=1; i<배열의 크기; i++)
```

```
    if ( prices[i] < minimum )
```

```
        minimum = prices[i]
```

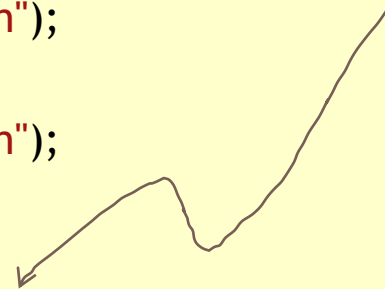
반복이 종료되면 minimum에 최소값이 저장된다.

최소값 찾기

38

```
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 10
int main(void)
{
    int prices[SIZE] = { 0 };
    int i, minimum;
    printf("-----\n");
    printf("1 2 3 4 5 6 7 8 9 10\n");
    printf("-----\n");
    srand( (unsigned)time( NULL ) );
    for(i = 0; i < SIZE; i++){
        prices[i] = (rand()%100)+1;
        printf("%-3d ",prices[i]);
    }
    printf("\n\n");
```

물건의 가격
출력



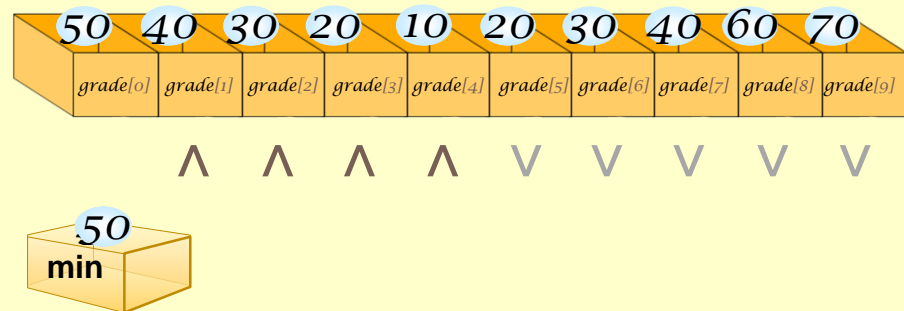
최소값 찾기

39

```
minimum = prices[0];  
for(i = 1; i < SIZE; i++)  
{  
    if( prices[i] < minimum )  
        minimum = prices[i];  
}  
printf("최소값은 %d입니다.\n", minimum);  
return 0;  
}
```

첫 번째 배열 원소를 최소
값으로 가정

현재의 최소값보다
배열 원소가
작으면, 배열 원
소를 최소값으로
복사한다.



도전문제

40

- 위의 프로그램에서는 최소값을 계산하였다. 이번에는 배열의 원소 중에서 최대값을 찾으도록 변경하여 보자. 변수 이름도 적절하게 변경하라.



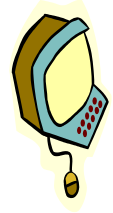
실습: 투표 집계하기

41

- 투표 결과를 컴퓨터를 이용하여서 집계한다고 가정하자.
- 데이터의 빈도(frequency)를 계산하는 것과 동일
- 배열의 개념을 이용하면 손쉽게 구현할 수 있다.

실행 결과

42



몇번 후보자를 선택하시겠습니까?(종료 -1): 1
몇번 후보자를 선택하시겠습니까?(종료 -1): 1
몇번 후보자를 선택하시겠습니까?(종료 -1): -1

값	득표결과
1	2
2	0
3	0
4	0
	...
9	0
10	0

알고리즘

43

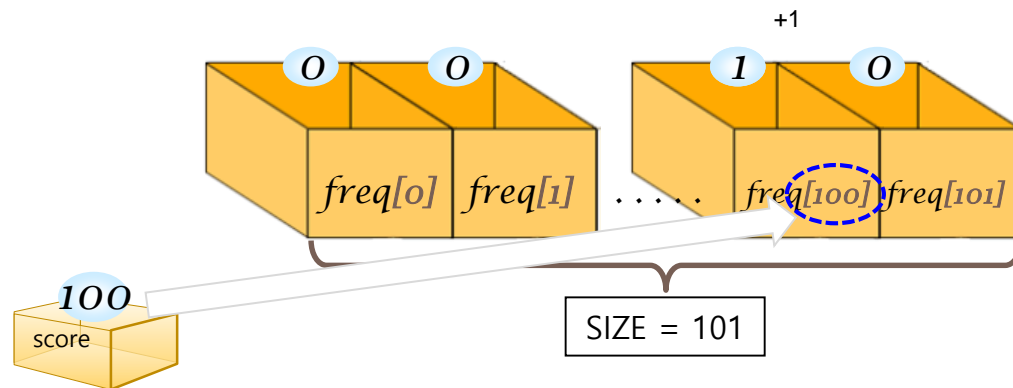
배열 `freq[]`의 원소를 0으로 초기화한다.

`while(1)`

 사용자로부터 후보자를 입력받는다.

`freq[candidate]++;`

`freq` 배열의 내용을 출력한다.





```
#include <stdio.h>
#define SIZE 11
int main(void)
{
    int freq[SIZE] = { 0 };
    int i, candidate;
    while(1)
    {
        printf("몇번 후보자를 선택하시겠습니까?(종료-1): ");
        scanf("%d", &candidate);
        if (candidate < 0) break;
        freq[candidate]++;
    }
    printf("값 득표결과\n");
    for(i = 1; i < SIZE; i++)
        printf("%3d %3d \n", i, freq[i]);
    return 0;
}
```

// 빈도를 나타내는 배열

// 음수이면 반복종료

해당되는 점수의 빈도를 증가한다

도전문제

45

- C에서는 배열의 인덱스가 항상 0부터 시작한다. 하지만 일상 생활에서는 번호가 1부터 시작한다. 여기서 발생하는 문제가 상당히 있다. 위의 프로그램에서도 만약 배열의 크기를 10으로 한다면 어떻게 변경하여야 하는가?



Contents

46

10.1

배열이란?

10.2

배열의 초기화

10.3

배열과 함수

10.4

정렬

10.5

탐색


10.6

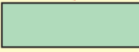
2차원 배열

배열과 함수

47

- 배열의 경우에는 사본이 아닌 원본이 전달된다.

```
int main(void)
{
    ...
    get_average(, STUDENTS);
    ...
}
```

```
int get_average(, int size)
{
    ...
    sum += scores[i];
    ...
}
```

배열 매개
변수의 경우,
원본이 직접
참조됩니다.



배열과 함수

48

```
#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;
}
```

배열이 인수인 경우,
참조에 의한 호출

```
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;
}
```

배열의 원본이
score[]로 전달

배열이 함수의 인수인 경우 1/2

49

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

void modify_array(int a[], int size);
void print_array(int a[], int size);

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

    print_array(list, SIZE);           // 배열은 원본이 전달된다. (인수 : 배열)
    modify_array(list, SIZE);
    print_array(list, SIZE);

    return 0;
}
```

배열이 함수의 인수인 경우 2/2

50

```
void modify_array(int a[], int size)
```

```
{  
    int i;  
  
    for(i = 0; i < size; i++)  
        a[i] = a[i] * a[i];  
}
```

```
void print_array(int a[], int size)
```

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

배열의 원본이
a[]로 전달



원본 배열의 변경을 금지하는 방법

51

```
void print_array(const int a[], int size)
```

```
{
```

```
    ...
```

```
    a[0] = 100;           // 컴파일 오류!
```

```
}
```

함수 안에서 a[]는 변경할 수 없다.

Contents

52

10.1

배열이란?

10.2

배열의 초기화

10.3

배열과 함수

10.4

정렬

10.5

탐색

10.6

2차원 배열

정렬이란?

53

- 정렬은 물건을 크기순으로 오름차순이나 내림차순으로 나열하는 것
- 정렬은 컴퓨터 공학분야에서 가장 기본적이고 중요한 알고리즘중의 하나



정렬이란?

54

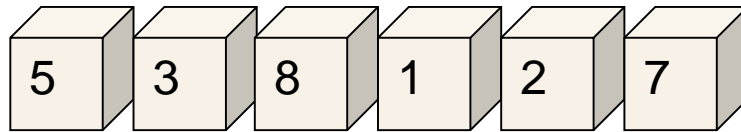
- 정렬은 자료 탐색에 있어서 필수적이다.
- (예) 만약 사전에서 단어들이 정렬이 안되어 있다면?



선택정렬(selection sort)

55

- 선택정렬(selection sort): 정렬이 안된 숫자들중에서 최소값을 선택하여 배열의 첫번째 요소와 교환
- 몇 개의 단계만 살펴보자.



선택정렬(selection sort)

56

- 선택정렬(selection sort): 정렬이 안된 숫자들중에서 최소값을 선택하여 배열의 첫번째 요소와 교환



선택 정렬(selection sort)

57

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

int main(void)
{
    int list[SIZE] = { 3, 2, 9, 7, 1, 4, 8, 0, 6, 5 };
    int i, j, temp, least;

    for(i = 0; i < SIZE-1; i++)
    {
        least = i;
        for(j = i + 1; j < SIZE; j++)
            if(list[j] < list[least])
                least = j;

        temp = list[i];
        list[i] = list[least];
        list[least] = temp;
    }

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

    printf("\n");
    return 0;
}
```

내부 **for** 루프로서 **(i+1)**번째 원소부터 배열의 마지막 원소 중에서 최소값을 찾는다. 현재의 최소값과 비교하여 더 작은 정수가 발견되면 그 정수가 들어 있는 인덱스를 **least**에 저장한다.

list[i]와 list[least]를 서로 교환



변수의 값을 서로 교환할 때

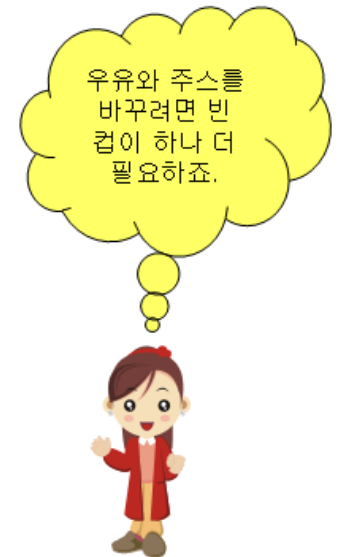
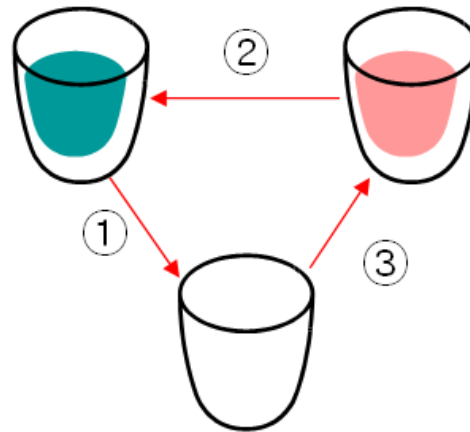
58

□ 다음과 같이 하면 안됨

- ▣ `grade[i] = grade[least];` // `grade[i]`의 기존값은 파괴된다!
- ▣ `grade[least] = grade[i];`

□ 올바른 방법

- ▣ `temp = list[i];`
- ▣ `list[i] = list[least];`
- ▣ `list[least] = temp;`



Contents

59

10.1

배열이란?

10.2

배열의 초기화

10.3

배열과 함수

10.4

정렬

10.5

탐색

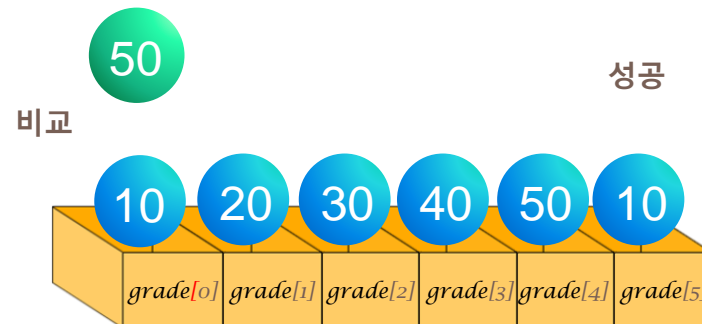
10.6

2차원 배열

순차탐색

60

- 순차 탐색은 배열의 원소를 순서대로 하나씩 꺼내서 탐색키와 비교하여 원하는 값을 찾아가는 방법



순차 탐색

61

```
#include <stdio.h>

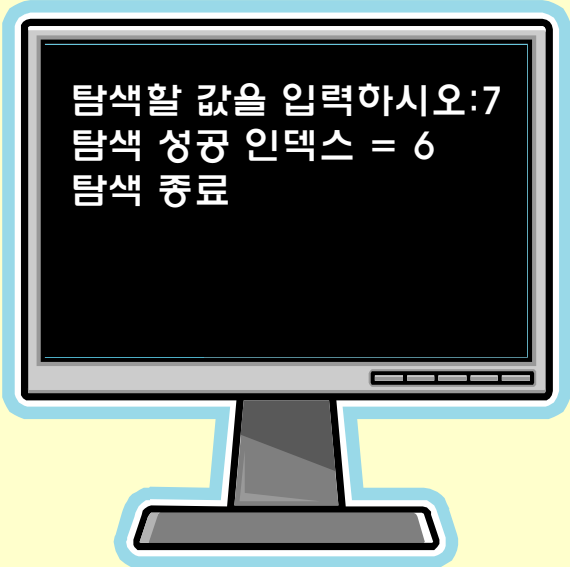
#define SIZE 10

int main(void)
{
    int key, i;
    int list[SIZE] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    printf("탐색할 값을 입력하시오:");
    scanf("%d", &key);

    for(i = 0; i < SIZE; i++)
        if(list[i] == key)

    printf("탐색 성공 인덱스= %d\n", i);
    printf("탐색 종료\n");
    return 0;
}
```

for 루프를 이용하여 list[i]와 key를 비교하는 연산을 배열의 크기만큼 반복한다. 만약 list[i]와 key가 같으면 탐색은 성공되고 키값이 발견된 배열의 인덱스를 출력한다.

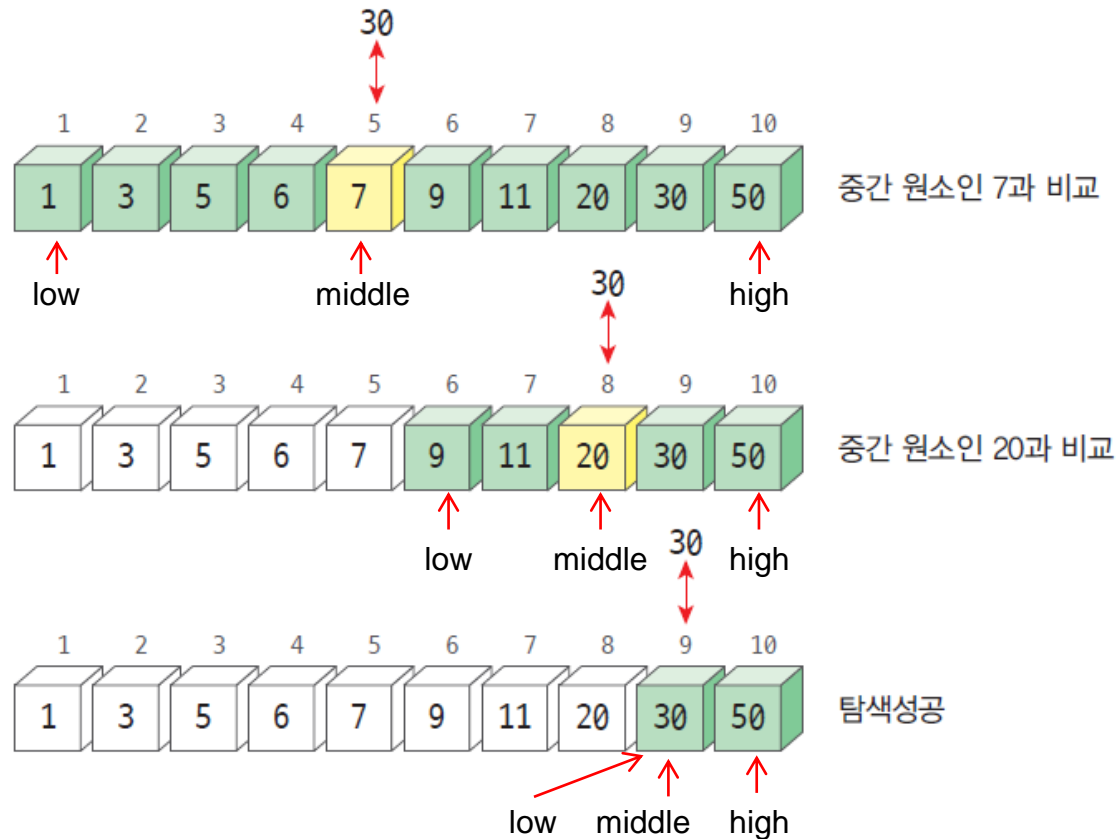


탐색할 값을 입력하시오:7
탐색 성공 인덱스 = 6
탐색 종료

이진 탐색

62

- 이진 탐색(binary search): 정렬된 배열의 중앙에 위치한 원소와 비교 되풀이



이진 탐색

63

```
#include <stdio.h>
#define SIZE 16
int binary_search(int list[], int n, int key);

int main(void)
{
    int key;
    int grade[SIZE] = { 2,6,11,13,18,20,22,27,29,30,34,38,41,42,45,47 };
    printf("탐색할 값을 입력하시오:");
    scanf("%d", &key);
    printf("탐색 결과= %d\n", binary_search(grade, SIZE, key));

    return 0;
}
```

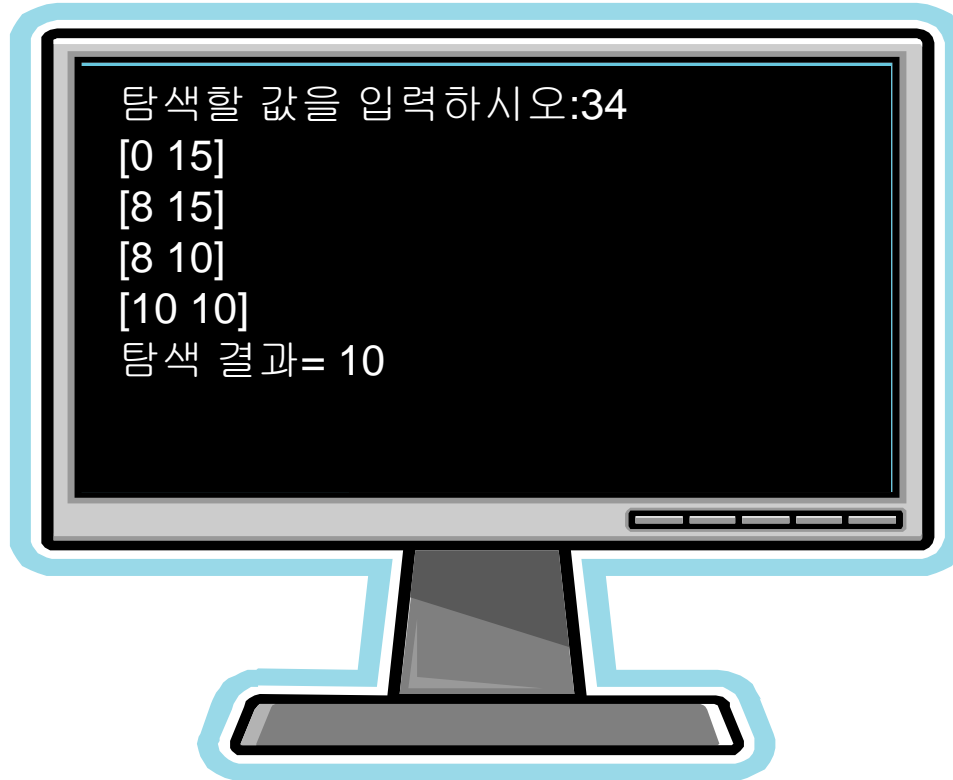
이진 탐색

64

```
int binary_search(int list[], int n, int key)
{
    int low, high, middle;
    low = 0;
    high = n-1;
    while( low <= high ){
        printf("[%d %d]\n", low, high); // 아직 숫자들이 남아있으면
        middle = (low + high)/2; // 하한과 상한을 출력한다.
        if( key == list[middle] ) // 중간 위치를 계산한다.
            return middle; // 일치하면 탐색 성공
        else if( key > list[middle] ) // 중간 원소보다 크다면
            low = middle + 1; // 새로운 값으로 low 설정
        else
            high = middle - 1; // 새로운 값으로 high 설정
    }
    return -1;
}
```


실행 결과

65



Contents

66

10.1

배열이란?

10.2

배열의 초기화

10.3

배열과 함수

10.4

정렬

10.5

탐색

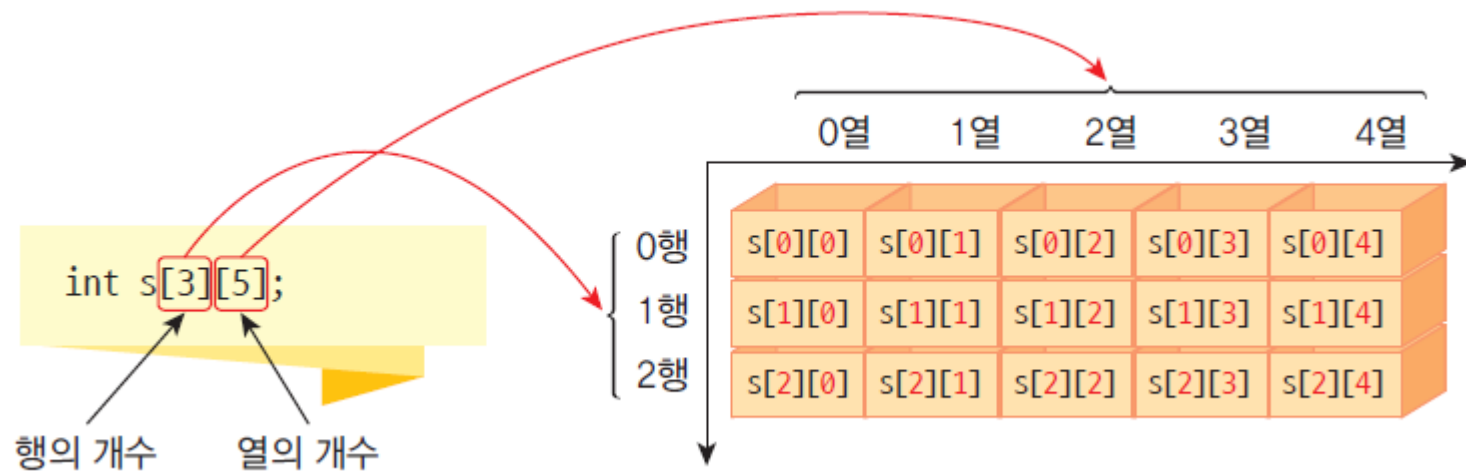
10.6

2차원 배열

2차원 배열

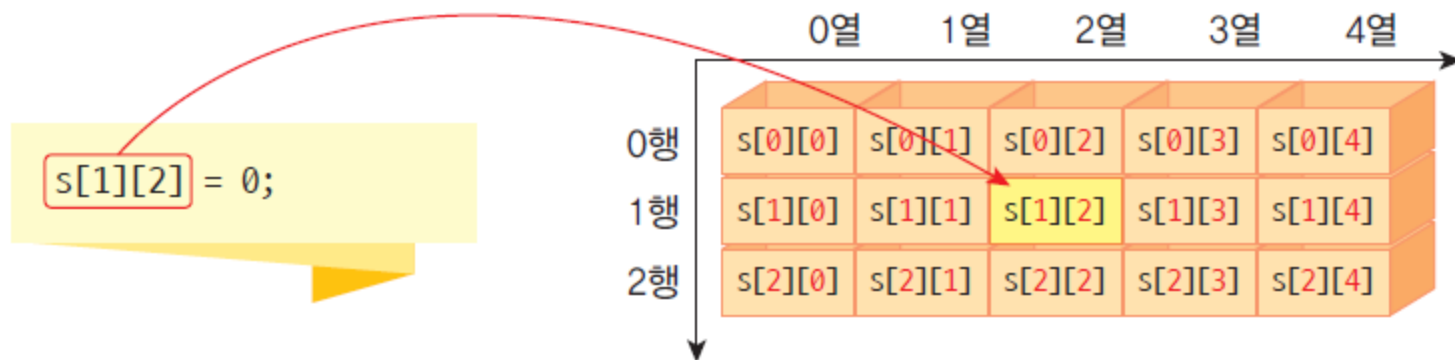
67

```
int s[10];    // 1차원 배열  
int s[3][10]; // 2차원 배열  
int s[5][3][10]; // 3차원 배열
```



2차원 배열에서 인덱스

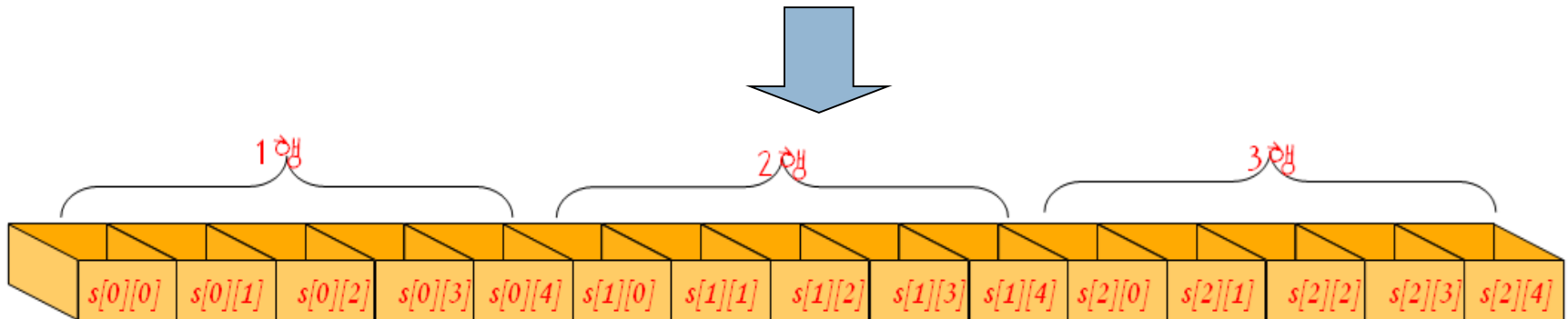
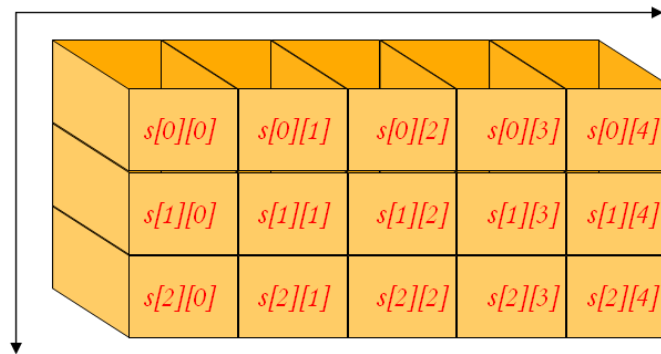
68



2차원 배열의 구현

69

- 2차원 배열은 1차원적으로 구현된다.



2차원 배열의 활용

70

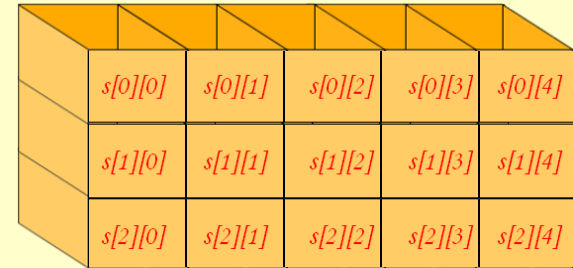
```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int s[3][5];           // 2차원 배열 선언
    int i, j;              // 2개의 인덱스 변수

    for(i=0;i<3;i++)
        for(j=0;j<5;j++)
            s[i][j] = rand()%100;

    for(i=0;i<3;i++)
        for(j=0;j<5;j++)
            printf("%02d", s[i][j]);
        printf("\n")

    return 0;
}
```



A 3D diagram representing a 3x5 array. It consists of three rows of five orange boxes. Each box is labeled with its row and column index in red text. The labels are: Row 0: s[0][0], s[0][1], s[0][2], s[0][3], s[0][4]; Row 1: s[1][0], s[1][1], s[1][2], s[1][3], s[1][4]; Row 2: s[2][0], s[2][1], s[2][2], s[2][3], s[2][4].

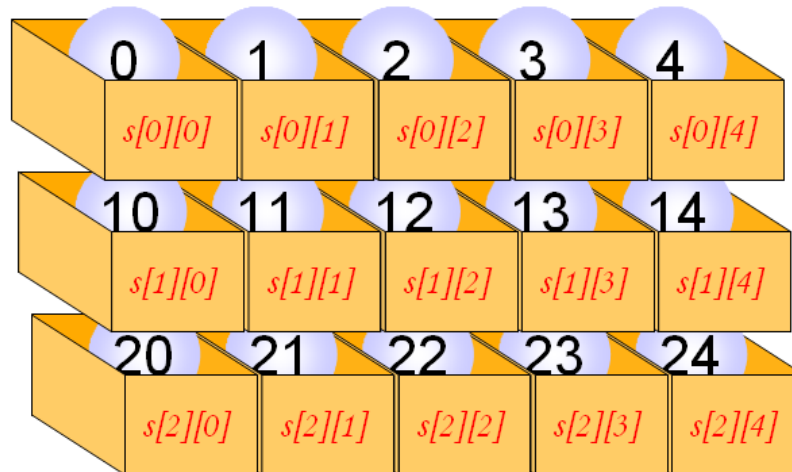
s[0][0]	s[0][1]	s[0][2]	s[0][3]	s[0][4]
s[1][0]	s[1][1]	s[1][2]	s[1][3]	s[1][4]
s[2][0]	s[2][1]	s[2][2]	s[2][3]	s[2][4]



2차원 배열의 초기화

71

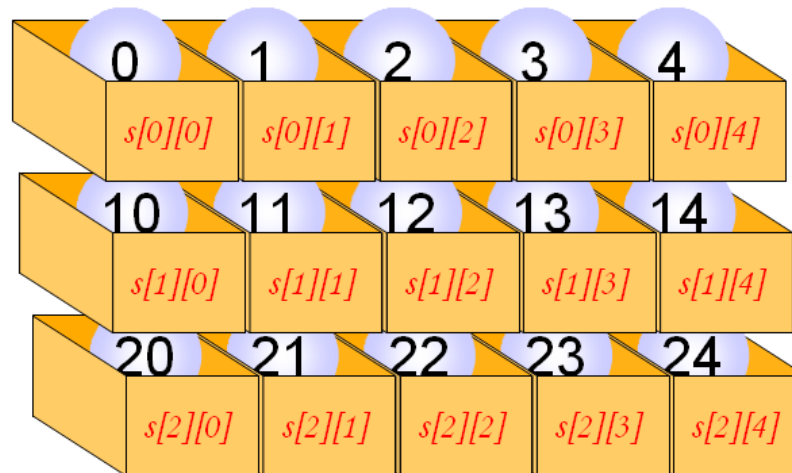
```
int s[3][5] = {  
    { 0, 1, 2, 3, 4 }, // 첫 번째 행의 원소들의 초기값  
    { 10, 11, 12, 13, 14 }, // 두 번째 행의 원소들의 초기값  
    { 20, 21, 22, 23, 24 } // 세 번째 행의 원소들의 초기값  
};
```



2차원 배열의 초기화

72

```
int s[ ][5] = {  
    { 0, 1, 2, 3, 4 }, // 첫 번째 행의 원소들의 초기값  
    { 10, 11, 12, 13, 14 }, // 두 번째 행의 원소들의 초기값  
    { 20, 21, 22, 23, 24 }, // 세 번째 행의 원소들의 초기값  
};
```



2차원 배열의 초기화

73

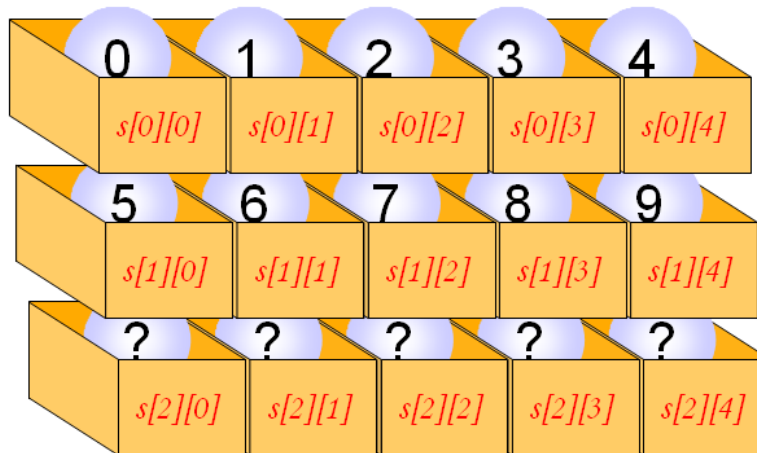
```
int s[ ][5] = {  
    { 0, 1, 2 },      // 첫 번째 행의 원소들의 초기값  
    { 10, 11, 12 },   // 두 번째 행의 원소들의 초기값  
    { 20, 21, 22 }    // 세 번째 행의 원소들의 초기값  
};
```



2차원 배열의 초기화

74

```
int s[ ][5] = {  
    0, 1, 2, 3, 4,    // 첫 번째 행의 원소들의 초기값  
    5, 6, 7, 8, 9,    // 두 번째 행의 원소들의 초기값  
};
```



3차원 배열

75

```
int s [6][3][5];
```

첫번째 두번째 세번째
인덱스: 인덱스: 인덱스:
학년번호 학급번호 학생번호

```
#include <stdio.h>
int main(void)
{
    int s[3][3][3];    // 3차원 배열 선언
    int x, y, z;        // 3개의 인덱스 변수
    int i = 1;          // 배열 원소에 저장되는 값

    for(z=0;z<3;z++)
        for(y=0;y<3;y++)
            for(x=0;x<3;x++)
                s[z][y][x] = i++;

    return 0;
}
```

예제

76

- 학생들의 성적 기록표를 2차원 배열에 저장하고 각 학생의 최종 성적을 계산해보자.

학번	중간고사(30%)	기말고사(40%)	기말과제(20%)	퀴즈점수(10%)	결석횟수(감점)
1	87	98	80	76	3
2	99	89	90	90	0
3	65	68	50	49	0

2차원 배열의 초기화

77

```
#include <stdio.h>
#define ROWS 3
#define COLS 5

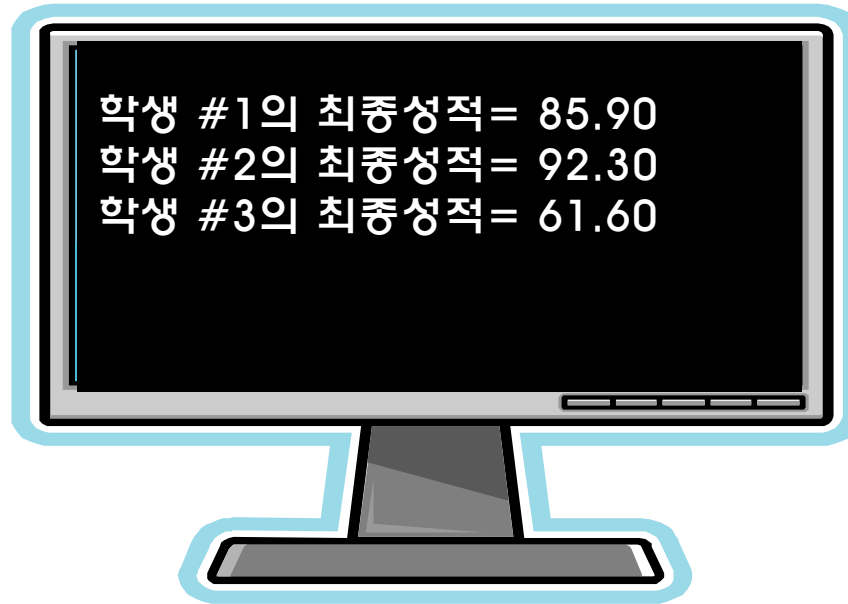
int main(void) {

    int a[ROWS][COLS] = { { 87, 98, 80, 76, 3 },
        { 99, 89, 90, 90, 0 },
        { 65, 68, 50, 49, 0 }
    };

    int i;
    for (i = 0; i < ROWS; i++) {
        double final_scores = a[i][0] * 0.3 + a[i][1] * 0.4 +
            a[i][2] * 0.2 + a[i][3] * 0.1 - a[i][4];
        printf("학생 #%i의 최종성적 = %10.2f \n", i + 1, final_scores);
    }
    return 0;
}
```

실행 결과

78

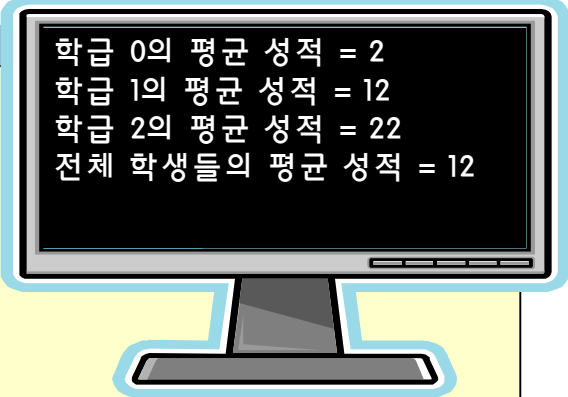


다차원 배열 예제

79

```
#include <stdio.h>
#define CLASSES 3
#define STUDENTS 5

int main(void)
{
    int s[CLASSES][STUDENTS] = {
        { 0, 1, 2, 3, 4 },    // 첫번째 행의 원소들의 초기값
        { 10, 11, 12, 13, 14 }, // 두번째 행의 원소들의 초기값
        { 20, 21, 22, 23, 24 }, // 세번째 행의 원소들의 초기값
    };
    int clas, student, total, subtotal;
    total = 0;
    for(clas = 0; clas < CLASSES; clas++)
    {
        subtotal = 0;
        for(student = 0; student < STUDENTS; student++)
            subtotal += s[clas][student];
        printf("학급 %d의 평균 성적= %d\n", clas, subtotal / STUDENTS);
        total += subtotal;
    }
    printf("전체 학생들의 평균 성적= %d\n", total/(CLASSES * STUDENTS));
    return 0;
}
```



학급 0의 평균 성적 = 2
학급 1의 평균 성적 = 12
학급 2의 평균 성적 = 22
전체 학생들의 평균 성적 = 12

- 행렬(matrix)은 자연과학에서 많은 문제를 해결하는데 사용

$$A = \begin{bmatrix} 2 & 3 & 0 \\ 8 & 9 & 1 \\ 7 & 0 & 5 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 7 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix}$$

Mathematics - ELEMENTARY MATRIX OPERATIONS

OPERATION: MULTIPLY EACH ELEMENT IN 2nd Row by 7:

$A = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$

1) FIND $E = 2 \times 2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix}$
 $I \quad E$

2) PREMULT $\begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} 1(0)+0(3) & 1(1)+0(4) & 1(2)+0(5) \\ 0(0)+7(3) & 0(1)+7(4) & 0(2)+7(5) \end{bmatrix}$

다차원 배열을 이용한 행렬의 표현

81

```
#include <stdio.h>
#define ROWS 3
#define COLS 3

int main(void)
{
    int A[ROWS][COLS] = { { 2,3,0 },
                          { 8,9,1 },
                          { 7,0,5 } };
    int B[ROWS][COLS] = { { 1,0,0 },
                          { 1,0,0 },
                          { 1,0,0 } };

    int C[ROWS][COLS];
    int r,c;
    // 두개의 행렬을 더한다.
    for(r = 0; r < ROWS; r++)
        for(c = 0; c < COLS; c++)
            C[r][c] = A[r][c] + B[r][c];
    // 행렬을 출력한다.
    for(r = 0; r < ROWS; r++)
    {
        for(c = 0; c < COLS; c++)
            printf("%d ", C[r][c]);
        printf("\n");
    }
    return 0;
}
```

중첩 for 루프를 이용하여 행렬 A의
각 원소들과 행렬의 B의 각 원소들을
서로 더하여 행렬 C에 대입한다.



2차원 배열을 함수로 전달하기

82

```
#include <stdio.h>
#define YEARS 3
#define PRODUCTS 5
```

```
int sum(int scores[][PRODUCTS]);
```

```
int main(void)
```

```
{
```

```
    int sales[YEARS][PRODUCTS] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
```

```
    int total_sale;
```

```
    total_sale = sum(sales);
```

복사본 전달

```
    printf("총매출은 %d입니다.\n", total_sale);
```

```
    return 0;
```

```
}
```

```
int sum(int scores[YEARS][PRODUCTS])
```

```
{
```

```
    int y, p;
```

```
    int total = 0;
```

```
    for(y = 0; y < YEARS; y++)
```

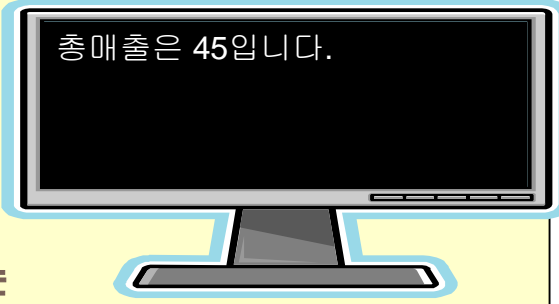
```
        for(p = 0; p < PRODUCTS; p++)
```

```
            total += scores[y][p];
```

```
    return total;
```

```
}
```

첫번째 인덱스의 크기는
적지 않아도 된다.



총매출은 45입니다.

Lab: 영상 처리

83

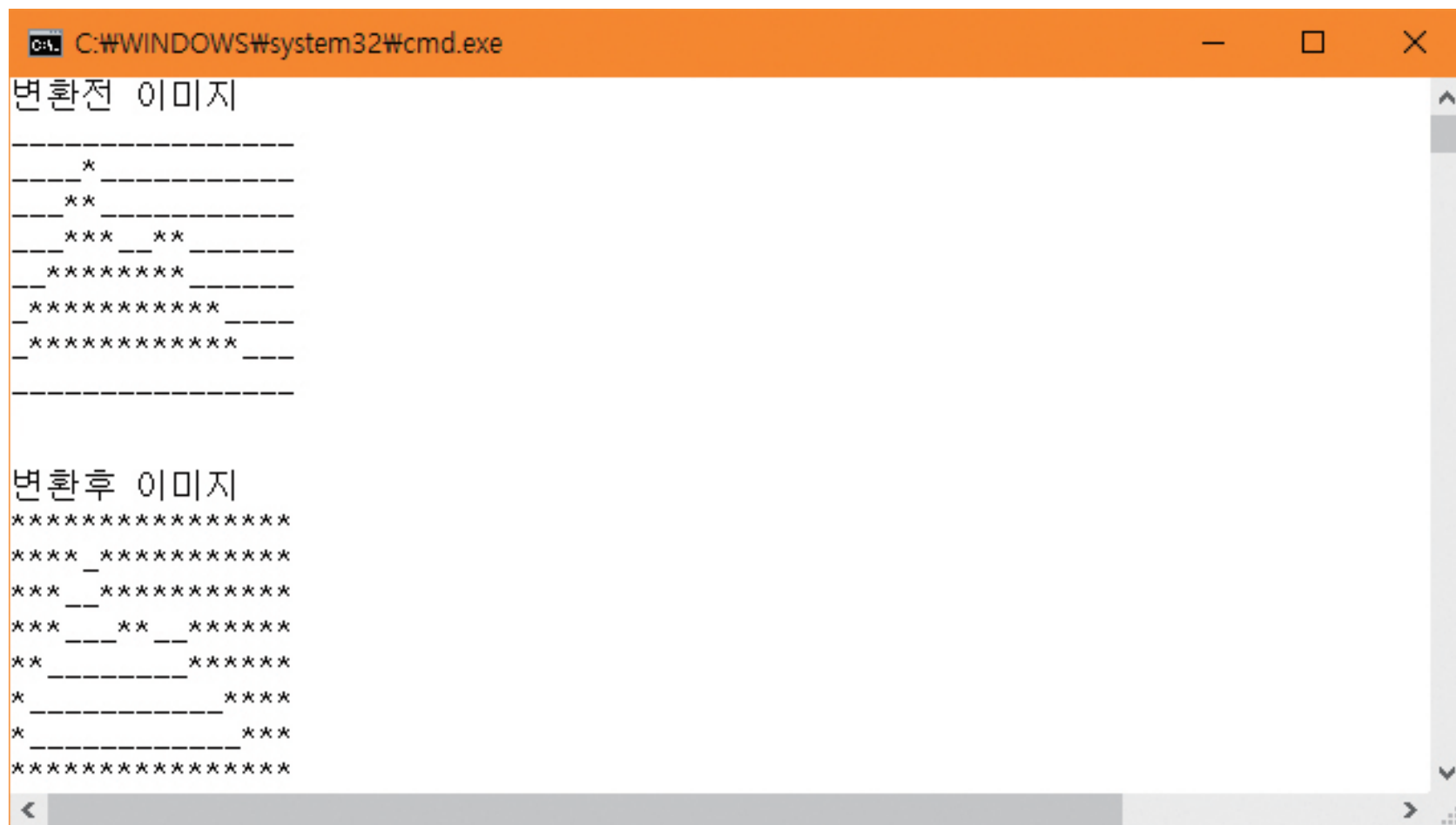
- 디지털 영상은 픽셀들의 2차원 배열이라 할 수 있다.

```
int image[8][16] = {  
    { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },  
    { 1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1 },  
    { 1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1 },  
    { 1,1,1,0,0,0,1,1,0,0,1,1,1,1,1,1 },  
    { 1,1,0,0,0,0,0,0,0,0,0,1,1,1,1,1 },  
    { 1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1 },  
    { 1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1 },  
    { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 } };
```



실행 결과

84



```
C:\WINDOWS\system32\cmd.exe

변환전 이미지
-----
      *
     **
    ***  **
   *****
  *****
 *****
-----

변환후 이미지
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
```

영상 처리

85

```
#include <stdio.h>
void display(int image[8][16]);
void inverse(int img[8][16]);

int main(void)
{
    int image[8][16] = {
        { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,0,0,0,1,1,0,0,1,1,1,1,1,1 },
        { 1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1 },
        { 1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1 },
        { 1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1 },
        { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 } };
    printf("변환전 이미지\n");
    display(image);
    inverse(image);
    printf("\n\n변환후 이미지\n");
    display(image);
    return 0;
}
```

영상 처리

86

```
void display(int image[8][16])
{
    for (int r = 0; r < 8; r++) {
        for (int c = 0; c < 16; c++) {
            if (image[r][c] == 0)
                printf("*");
            else
                printf("_");
        }
        printf("\n");
    }
}

void inverse(int img[8][16])
{
    for (int r = 0; r < 8; r++) {
        for (int c = 0; c < 16; c++) {
            if (img[r][c] == 0)
                img[r][c] = 1;
            else
                img[r][c] = 0;
        }
    }
}
```

Lab: tic-tac-toe

87

- tic-tac-toe 게임은 2명의 경기자가 오른쪽과 같은 보드를 이용하여서 번갈아가며 O와 X를 놓는 게임이다.
- 같은 글자가 가로, 세로, 혹은 대각선 상에 놓이면 이기게 된다.



실행 결과

88



(x, y) 좌표(종료 -1, -1): 0 0

---|---|---

X | |

---|---|---

| |

---|---|---

| |

---|---|---

(x, y) 좌표(종료 -1, -1): 1 1

---|---|---

X | |

---|---|---

| O |

---|---|---

| |

---|---|---

...

알고리즘

89

보드를 초기화한다.

while(1)

 보드를 화면에 출력한다.

 사용자로부터 좌표 x, y를 받는다.

 if (board[x][y]가 비어 있으면)

 if(현재 경기자가 'X'이면)

 board[x][y] = 'X'

 else

 board[x][y] = 'O'

 else

 오류 메시지를 출력한다



```
#include <stdio.h>
#include <stdlib.h>
void init_board(char board[][3]);
int get_player_move(int palyer, char board[][3]);
void disp_board(char board[][3]);
int main(void)
{
    char board[3][3];
    int quit=0;
    init_board(board);
    do {
        disp_board(board);
        quit = get_player_move(0, board);
        disp_board(board);
        quit = get_player_move(1, board);
    } while(quit == 0);
    return 0;
}
```



```
void init_board(char board[][3])
{
    int x, y;
    for(x=0; x<3; x++)
        for(y=0; y<3; y++) board[x][y] = ' ';
}
void disp_board(char board[3][3])
{
    int i;
    for(i=0; i<3; i++){
        printf("--- | --- | ---\n");
        printf(" %c | %c | %c \n",board[i][0], board[i][1], board [i][2]);
    }
    printf("--- | --- | ---\n");
}
```



```
int get_player_move(int player, char board[3][3])
{
    int x, y, done = 0;

    while(done != 1) {
        printf("(x, y) 좌표(종료-1, -1): ");
        scanf("%d %d", &x, &y);
        if( x == -1 && y == -1 ) return 1;
        if(board[x][y]==' ') break; // 올바른 위치이면 반복 루프를 나간다.
        else printf("잘못된 위치입니다.\n");
    }

    if( player == 0 )    board[x][y] = 'X';
    else board[x][y] = 'O';

    return 0;
}
```

도전문제

93

- 보드를 분석하여서 게임이 종료되었는지를 검사하는 함수를 추가하라.
- 컴퓨터가 다음 수를 결정하도록 프로그램을 변경하라. 가장 간단한 알고리즘을 사용한다. 예를 들면 비어 있는 첫 번째 좌표에 놓는다.

