

<자료구조 및 실습>

1. C 프로그래밍 복습

한국외국어대학교
컴퓨터.전자시스템공학전공
2016년 1학기
고 석 훈

학습 목표

- 배열
- 포인터
- 매개변수
- 구조체

배열(Array)

- 100명의 학생에 대한 성적을 나타내고자 한다면?

```
int  grade1, grade2, grade3, ..., grade100;  
int  grade[100];
```

- 배열(array)



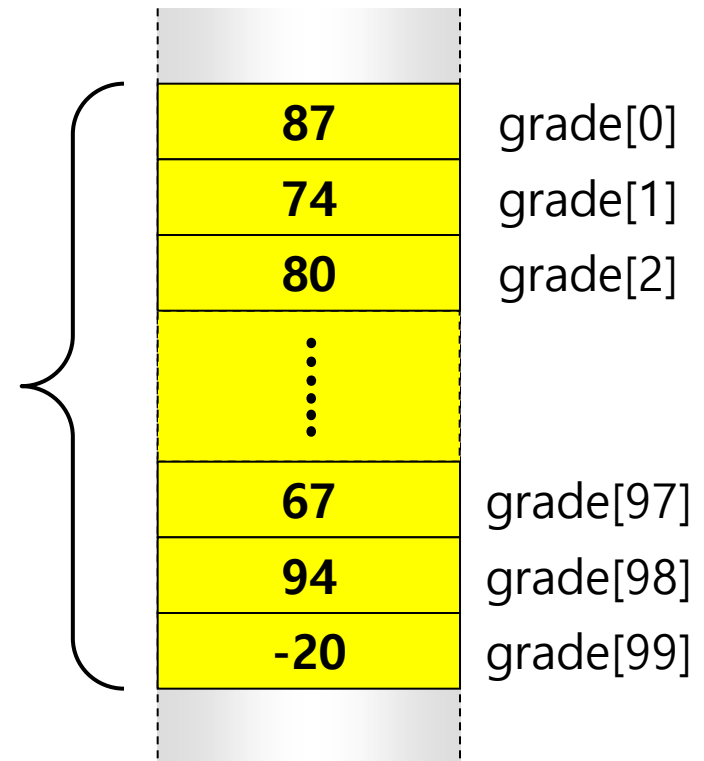
- 인덱스를 이용하여 데이터의 위치를 계산하여 액세스
- 예) 무지개 아파트 C동: 배열 이름
무지개 아파트 C동 803호: 개별적인 변수의 이름

int grade[100];

● 기억공간 할당

- 배열의 원소는 0번째부터 시작: grade[0], ..., grade[99]
- i-번째 배열 원소는 grade[i]로 참조

컴파일러는 100개의
4byte 정수형 변수를
grade란 이름으로
메모리에 할당한다



배열의 선언 및 초기화 [1/2]

```
#define N 3

void main(void)
{
    int a[N] = { 10, 3, 7 };
    int k, sum = 0;

    for (k = 0; k < N; k++)
        sum = sum + a[k];

    printf("평균=%d\n", sum/N);
}
```

```
#define N 3

void main(void)
{
    int a[] = { 10, 3, 7 };
    int k, sum = 0;

    for (k = 0; k < N; k++)
        sum = sum + a[k];

    printf("평균=%d\n", sum/N);
}
```

배열의 선언 및 초기화 [2/2]

● 배열의 선언

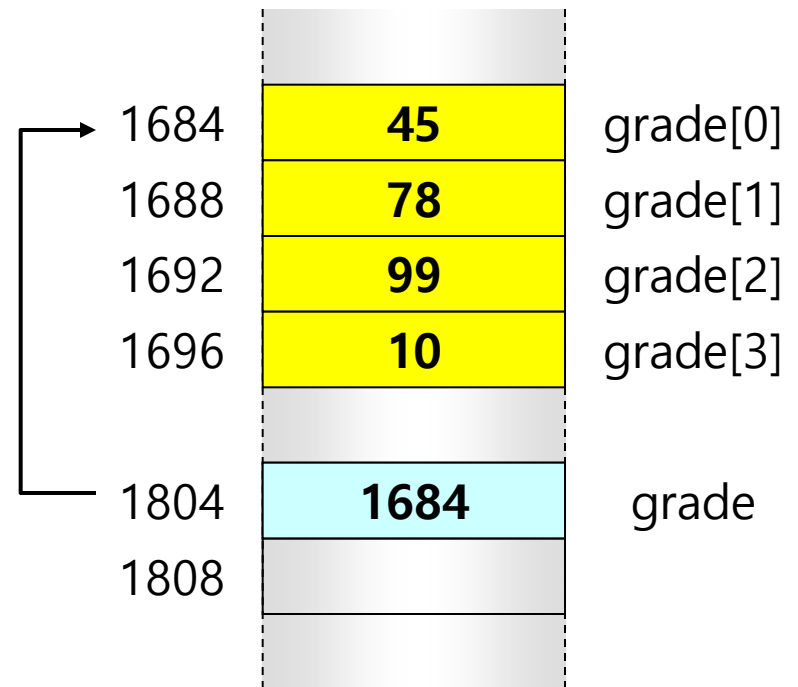
- `char s[100];`
- 100개의 문자 변수로 배열을 이룬 `s`라는 이름의 문자배열

● 초기화

- `char s1[] = "Korea";`
- `char s2[10] = {'K', 'o', 'r', 'e', 'a', '\0'};`
- `int a1[10] = { 0 };`
- `int a2[10] = { 0, 2, 10, 1, 2, 3, };`
- `int a3[10] = { 0, 2, 10, 1 };`

포인터(Pointer)

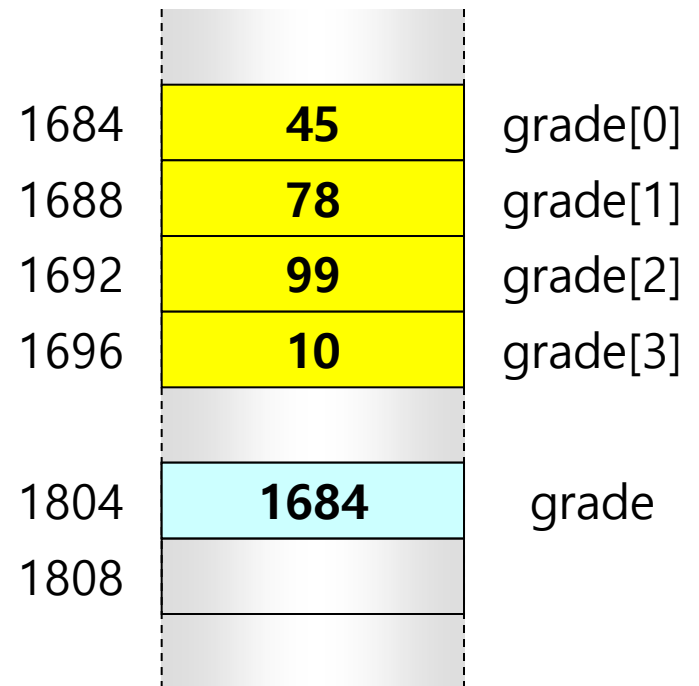
- `int grade[4] = { 45, 78, 99, 10 };`
 - `grade[0] ~ grade[3]`은 각각 정수형 변수를 의미한다.
- `grade`는 무엇을 의미하나?
 - 네 개의 정수형 변수로 구성된 배열의 이름
- 그럼, `grade`의 값은?
 - 배열의 시작 주소를 값으로 갖는다.
 - 즉, `grade[0]`를 포인팅 한다.



& 연산자

```
int grade[4] = { 45, 78, 99, 10 };
```

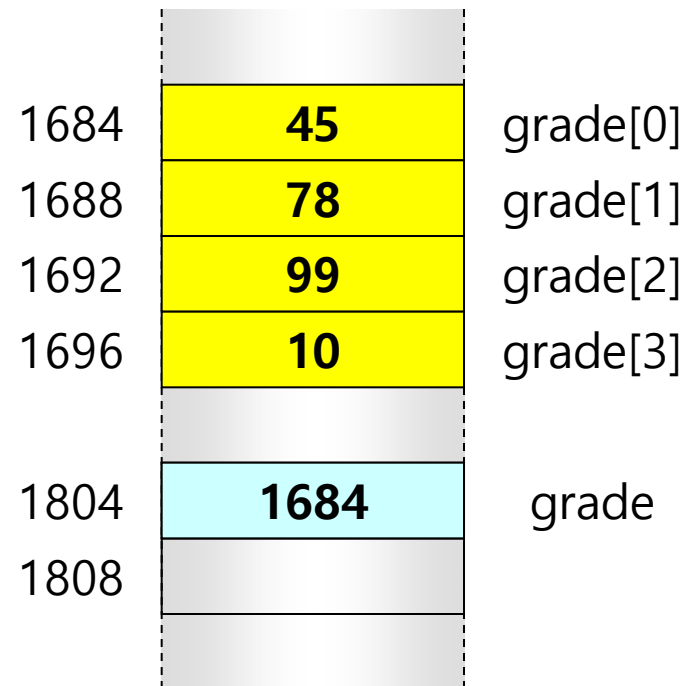
```
printf("%d\n", grade);  
printf("%d\n", grade[0]);  
printf("%d\n", &grade[0]);  
printf("%d\n", grade + 1);  
printf("%d\n", &grade[1]);
```



* 연산자

```
int grade[4] = { 45, 78, 99, 10 };
```

```
printf("%d\n", grade);  
printf("%d\n", grade[0]);  
printf("%d\n", *grade);  
printf("%d\n", *(grade + 1));  
printf("%d\n", *(&grade[1]));
```



포인터 변수 [1/2]

● `int *p;`

```
int grade[4] = { 45, 78, 99, 10 };
int *p = grade;
printf("%d\n", p);
printf("%d\n", *(grade + 1));
printf("%d\n", *(p + 1));
printf("%d\n", grade[1]);
printf("%d\n", &p);
```

1684	45	grade[0]
1688	78	grade[1]
1692	99	grade[2]
1696	10	grade[3]
1804	1684	grade
1808	1684	p

포인터 변수 [2/2]

```
int grade[4] = { 45, 78, 99, 10 };  
int *p, *q;
```

```
p = grade;  
q = grade + 2;  
printf("%d\n", *p - *q);  
q = p + 1;  
printf("%d\n", *q);  
printf("%d\n", p++);  
printf("%d\n", p + q);
```

1684	45	grade[0]
1688	78	grade[1]
1692	99	grade[2]
1696	10	grade[3]
1804	1684	grade
1808	1688	p
1812	1688	q

배열과 포인터

```
#define N      3

void main(void)
{
    int  a[N] = { 10, 3, 7 };
    int  k;
    int  sum = 0;

    for (k = 0; k < N; k++)
        sum = sum + a[k];

    printf("평균=%d\n", sum/N);
}
```

```
#define N      3

void main(void)
{
    int  a[N] = { 10, 3, 7 };
    int  *p = a;
    int  sum = 0;

    for (k = 0; k < N; k++)
        sum = sum + *(p + k);

    printf("평균=%d\n", sum/N);
}
```

매개변수 리스트(Parameter List)

- 매개변수 리스트(parameter list)는 함수를 호출한 블록과 호출된 함수간의 자료 공유를 위해 사용된다.
- 인자(argument)와 매개변수(parameter)

```
int func(int a, int b); // a, b is parameters  
ret = func(10, 2);     // 10, 2 is arguments
```

값 매개변수(Value Parameter)

-
- Call-by-Value 혹은 Pass-by-Value라고 부른다.
- 인자(argument)는 변수, 상수, 혹은 식이 될 수 있다.

```
int func(int a, int b);  
  
x = func(10, m);  
y = func(0, i + j);
```

예제) 값 매개변수

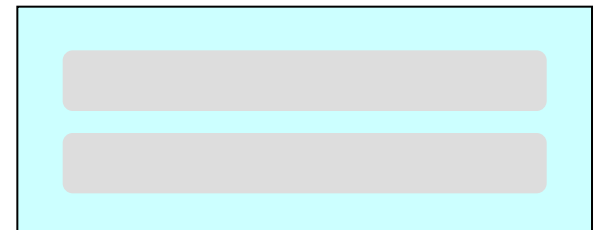
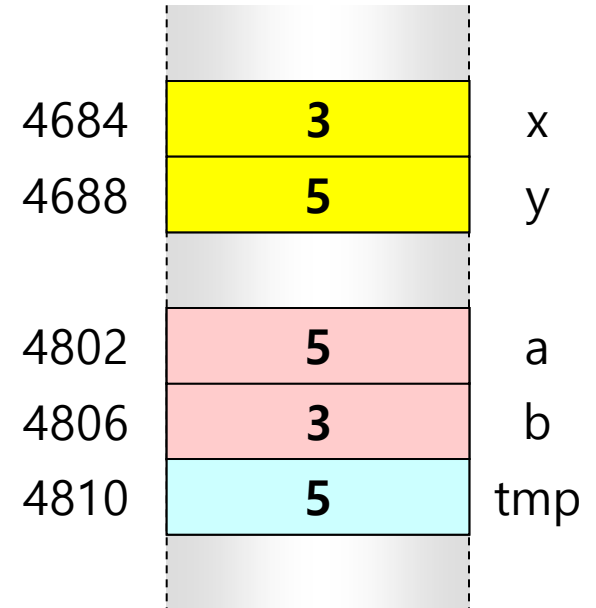
```
#include <stdio.h>

void swap(int a, int b)
{
    int tmp = b;

    b = a;
    a = tmp;
}

void main(void)
{
    int x = 3;
    int y = 5;

    printf("Before swap %d, %d\n", x, y);
    swap(x, y);
    printf("After swap %d, %d\n", x, y);
}
```



참조 매개변수(Reference Parameter)

-
- Call-by-Reference 혹은 Pass-by-Reference라 부른다.
- 인자(argument)는 항상 변수이어야 한다.

```
int func(int *a, int *b);  
int i, j, *p, *q;  
  
x = func(&i, &j);  
y = func(p, q);
```


예제) 참조 매개변수

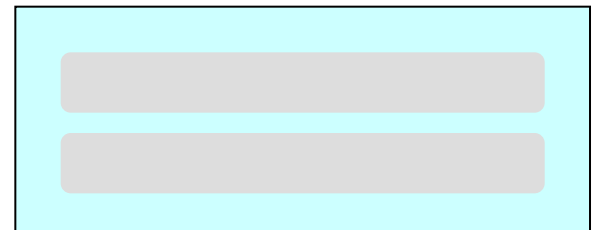
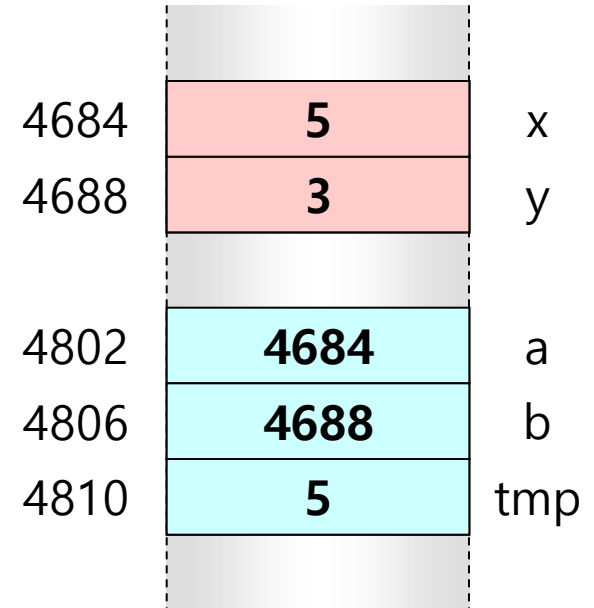
```
#include <stdio.h>

void swap(int *a, int *b)
{
    int tmp = *b;

    *b = *a;
    *a = tmp;
}

void main(void)
{
    int x = 3;
    int y = 5;

    printf("Before swap %d, %d\n", x, y);
    swap(&x, &y);
    printf("After swap %d, %d\n", x, y);
}
```



배열을 함수 인자로 전달

● 배열의 시작주소만 전달

- 배열 원소의 데이터와 배열의 크기 정보는 전달되지 않음

```
double sum1(double a[], int n);
double sum2(double *a, int n);

void main(void)
{
    double weight[4];
    double s1, s2;
    . . .
    s1 = sum1(weight, 4);
    s2 = sum2(weight, 4);
    . . .
}
```

1480	56.9	weight[0]
1488	68.7	weight[1]
1496	48.3	weight[2]
1504	87.5	weight[3]
1804	1480	weight
1808	1480	a

문자열(String)

● 문자열 변수

- 문자열의 내용과 '\0'(null 문자)를 메모리에 저장하고, 문자형 포인터 변수에 문자열의 주소를 배정

● 문자열 상수

- 문자열 상수는 배열과 마찬가지로 문자열을 메모리에 저장하고, 문자열의 주소를 의미

```
char *s1 = "tell me";  
char s2[] = "pineapple";
```

1684	't'	'e'	'l'	'l'	
1688	' '	'm'	'e'	'\0'	
1692	'p'	'i'	'n'	'e'	
1696	'a'	'p'	'p'	'l'	
1700	'e'	'\0'			
1704					
1708	1684				s1
1712	1692				s2

문자열 출력

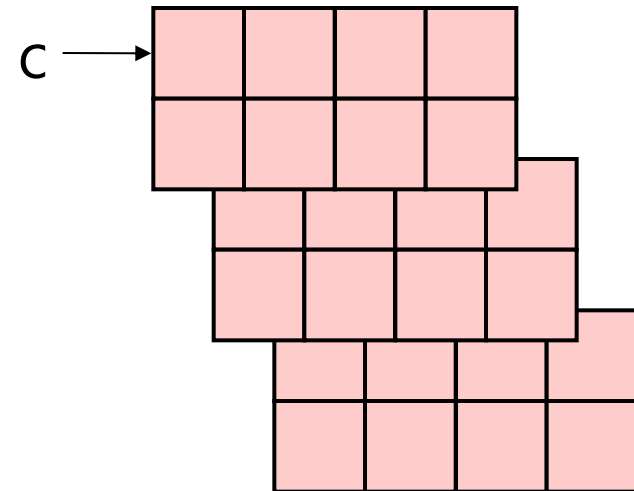
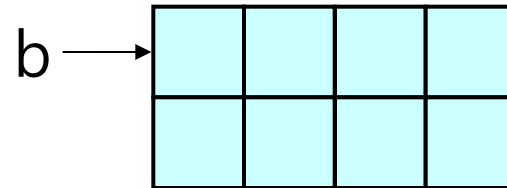
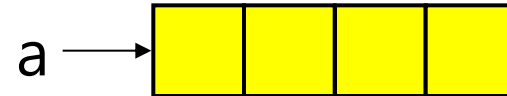
```
char *s1 = "tell me";
char s2[] = "pineapple";

printf("%s\n", s1);
printf("%s\n", s1+5);
printf("%c\n", *(s2+2));
printf("%c\n", s2[2]);
printf("%s\n", &s2[4]);
printf("%d\n", s2);
printf("%d\n", s2 + 4);
printf("%c\n", "wine"[3]);
printf("%c\n", *("wine"+3));
printf("%s\n", "wine"+1);
```

1684	't'	'e'	'l'	'l'	
1688	' '	'm'	'e'	'\0'	
1692	'p'	'i'	'n'	'e'	
1696	'a'	'p'	'p'	'l'	
1700	'e'	'\0'	'w'	'i'	
1704	'n'	'e'	'\0'		
1708	1684				s1
1712	1692				s2

다차원 배열

- `int a[4];`
 - 일차원 정수 배열
 - 4개의 정수변수로 구성
- `int b[2][4];`
 - 이차원 정수 배열
 - 2개의 행과 4개의 열로 구성
- `int c[3][2][4];`
 - 삼차원 정수 배열
 - 3 x 2 x 4 개의 정수로 구성



2차원 배열 [1/4]

● `int a[3][5];`

- `a`는 `a[0]`, `a[1]`, `a[2]` 세 원소로 구성된 1차원 배열이다.
- `a[0]`, `a[1]`, `a[2]` 는 다시 5개의 정수로 구성된 1차원 배열이다.

	0열	1열	2열	3열	4열
0행	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>	<code>a[0][4]</code>
1행	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>	<code>a[1][4]</code>
2행	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>	<code>a[2][4]</code>

● `a[i][j]`

- 2차원 배열 `a`의 `i`번째 행의 `j`번째 열의 원소, `(a[i])[j]` 와 동일
- `a[2][3]` : 1차원 배열 `a[2]`에서 3번째 원소를 의미

2차원 배열 [2/4]

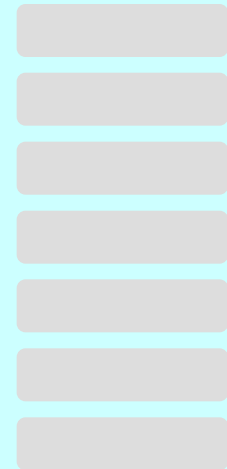
- 배열 이름 `a` 는 어떤 주소를 갖는가?
- `a[0]`, `a[1]`, `a[2]` 는 어떤 주소를 갖는가?

<code>a[0]</code>	4100	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>	<code>a[0][4]</code>
<code>a[1]</code>	4120	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>	<code>a[1][4]</code>
<code>a[2]</code>	4140	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>	<code>a[2][4]</code>
<code>a</code>	4160	4100				

2차원 배열 [3/4]

a[0]	4100	4	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1]	4120	a[1][0]	a[1][1]	10	a[1][3]	a[1][4]
a[2]	4140	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]
a	4160	4100				

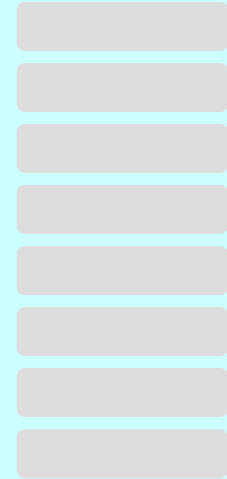
```
printf("%d\n", a);  
printf("%d\n", a[0]);  
printf("%d\n", a[1]);  
printf("%d\n", a[2]);  
printf("%d\n", a[0][0]);  
printf("%d\n", &a[0][0]);  
printf("%d\n", &a[1][2]);
```



2차원 배열 [4/4]

a[0]	4100	4	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1]	4120	a[1][0]	12	10	8	a[1][4]
a[2]	4140	a[2][0]	15	35	55	a[2][4]
a	4160	4100				

```
printf("%d\n", a + 1);  
printf("%d\n", a[1]);  
printf("%d\n", *(a + 1));  
printf("%d\n", a[0] + 1);  
printf("%d\n", a[1][2]);  
printf("%d\n", *(a[1] + 3));  
printf("%d\n", (*(a + 2))[1]);  
printf("%d\n", (*(a + 2) + 2));
```



typedef을 이용한 자료형 정의

```
#define MAX 3
typedef double scalar;
typedef scalar vector[MAX];
typedef vector matrix[MAX];
```

포인터 배열 [1/2]

● 정수 1차원 배열

- `int a[4];`

● 문자 1차원 배열

- `char b[5];`

● 실수 1차원 배열

- `double c[6];`

● 포인터 1차원 배열

- `int *a[4];`

- `char *b[5];`

- `double *c[6];`

포인터 배열 [2/2]

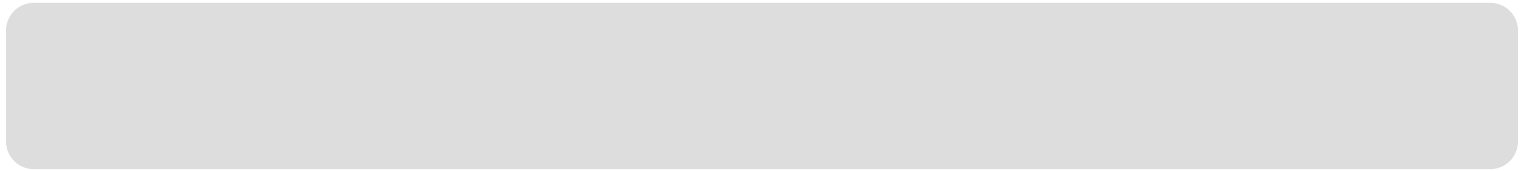
```
int  *a[3];  
int  b0 = 0, b1 = 1, b2 = 2;  
  
a[0] = &b2;  
a[1] = &b1;  
a[2] = &b0;  
printf("%d\n", a);  
printf("%d\n", a[0]);  
printf("%d\n", *a[0]);  
printf("%d\n", *(a+1));  
printf("%d\n", **(a+1));
```

1480	1504	a[0]
1484	1500	a[1]
1488	1496	a[2]
1492	1480	a
1496	0	b0
1500	1	b1
1504	2	b2

동적(dynamic) 메모리 할당

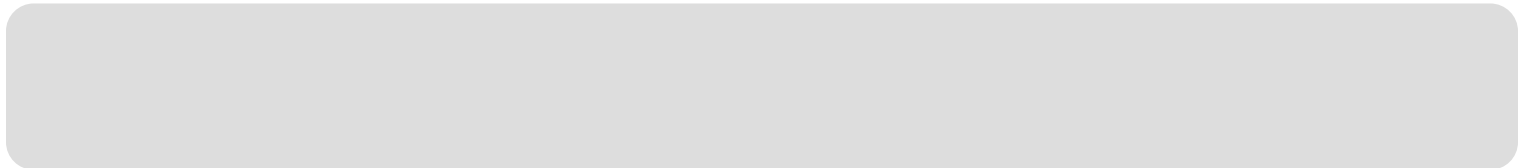
- 배열의 크기를 아는 경우

- `int a[5];`



- 배열의 크기를 미리 알 수 없는 경우

- `int *a;`
`a = (int*) malloc(n*sizeof(int));`



malloc 함수

- malloc (memory allocation)

- `void *malloc(size_t n)`



- 예제:

- `int *a;`
`a = (int*) malloc(n*sizeof(int));`

구조체 [1/3]

- 예) 주소록 선언

- 각각의 항목을 변수로 선언

```
char  name[20];  
char  phone[20];  
char  address[60];  
int    birthday;
```

- 100명의 주소록을 선언하는 방법?

- 각 항목을 배열로 선언

```
char  name[100][20];  
char  phone[100][20];  
char  address[100][60];  
int    birthday[100];
```

구조체 [2/3]

- "김건모"의 전화번호를 찾고 싶다면?

```
for (i = 0; i < 100; i++) {  
    if (strcmp(name[i], "김건모") == 0) {  
        printf(" phone = %d \n", phone[i]);  
        break;  
    }  
}
```

- "장동건"의 전화번호와 주소를 바꾸고 싶다면?

```
for (i = 0; i < 100; i++) {  
    if (strcmp(name[i], "장동건") == 0) {  
        strcpy(phone[i], "0313304640");  
        strcpy(address[i], "서울역 광장");  
        break;  
    }  
}
```


구조체 [3/3]

● 구조체 (Struct)

- 서로 다른 형의 변수들을 하나로 묶어주는 방법
- 예) 이름, 전화번호, 주소, 생일을 묶어 명함으로 관리
- 이때, name[20], phone[20], address[60], birthday는 name_card 구조체의 멤버라 한다.

```
struct name_card {  
    char name[20];  
    char phone[20];  
    char address[60];  
    int birthday;  
};
```

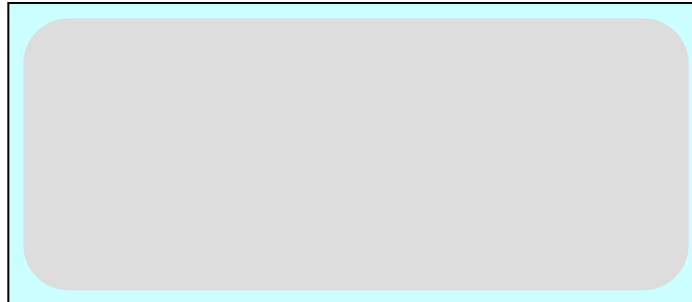
구조체 멤버 [1/2]

● struct name_card friend;

- friend 란 변수를 구조체 name_card 자료형으로 선언

friend {
 char name[20];
 char phone[20];
 char address[60];
 int birthday;

- friend 변수의 멤버에는 어떻게 접근하는가?



구조체 멤버 [2/2]

- struct name_card friend;
 - friend 변수에 "김건모", "2242424", "모현면 왕산리", 19690319 라는 정보를 입력하고 싶다면?

```
strcpy(friend.name, "김건모");  
strcpy(friend.phone, "02242424");  
strcpy(friend.address, "모현면 왕산리");  
friend.birthday = 19690319;
```

구조체 선언 [1/2]

```
struct name_card {  
    char  name[20];  
    char  phone[20];  
    char  address[60];  
    int    birthday;  
};  
  
struct name_card  friend, girl;
```

```
struct name_card {  
    char  name[20];  
    char  phone[20];  
    char  address[60];  
    int    birthday;  
} friend, girl;
```

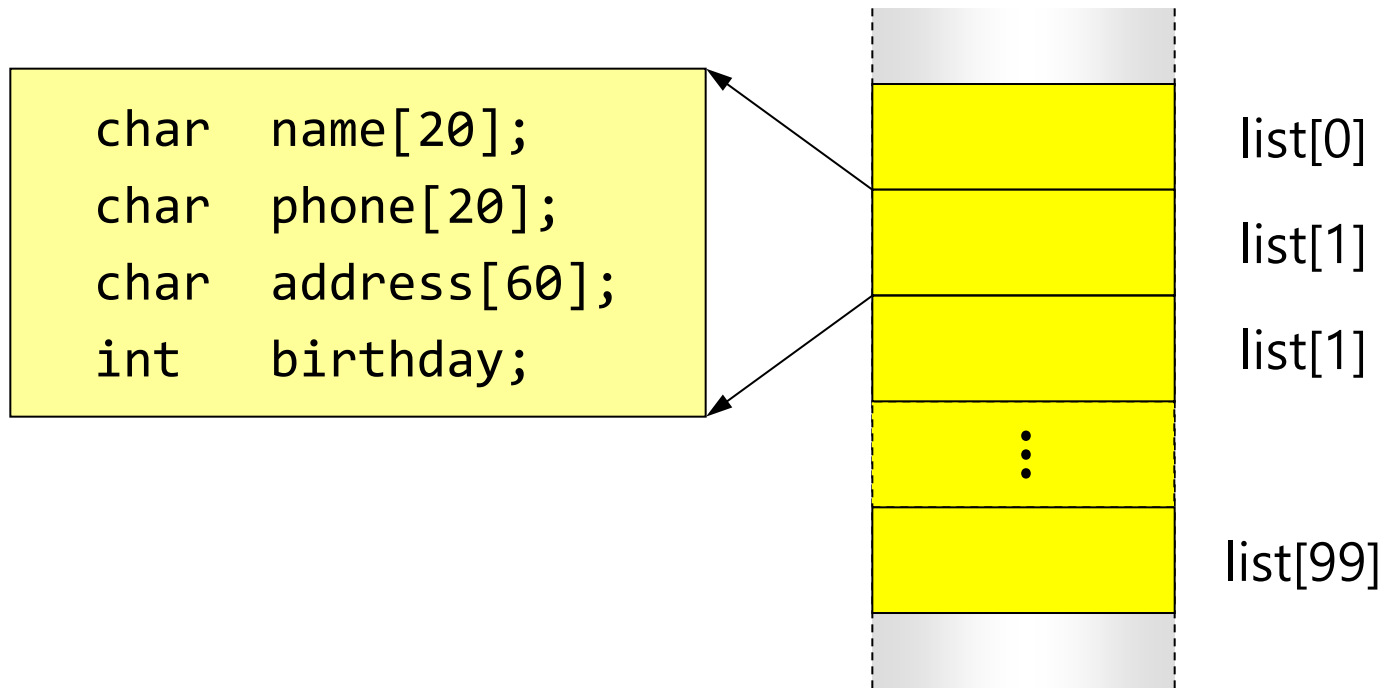
구조체 선언 [2/2]

```
struct name_card {  
    char  name[20];  
    char  phone[20];  
    char  address[60];  
    int    birthday;  
};  
  
typedef struct name_card CardType;  
  
CardType friend, girl;
```

```
typedef struct name_card {  
    char  name[20];  
    char  phone[20];  
    char  address[60];  
    int    birthday;  
} CardType;  
  
CardType friend, girl;
```

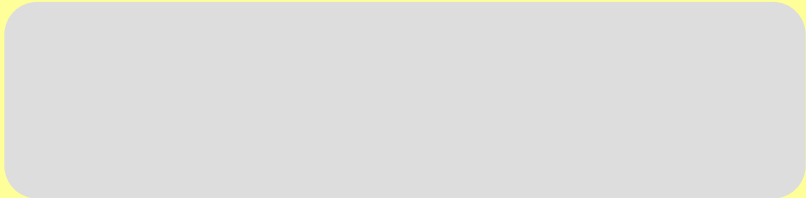
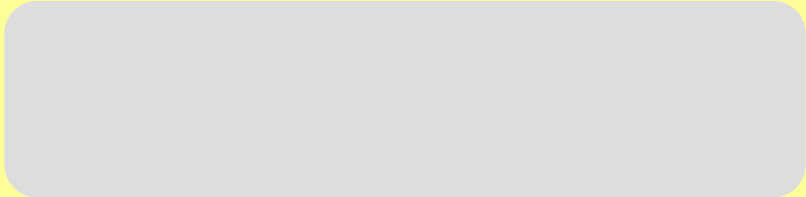
구조체 배열

- `struct name_card list[100];`



중첩 구조체

```
struct name_card {  
    char name[20];  
    char phone[20];  
    char address[60];  
    int birthday;  
};
```

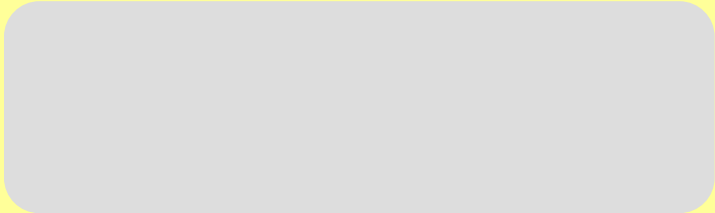
```
struct name_card {  
    char name[20];  
  
    char address[60];  
  
};
```

중첩 구조체의 멤버접근

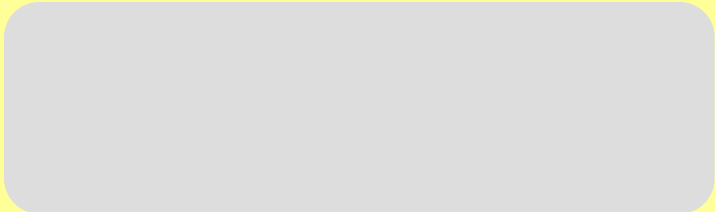
```
struct name_card {  
    char    name[20];  
    struct phone_info {  
        int reg, first, second;  
    } phone;  
    char    address[60];  
    struct birth_info {  
        int year, month, date;  
    } birthday;  
};
```

```
struct name_card girl;
```

```
girl.name
```

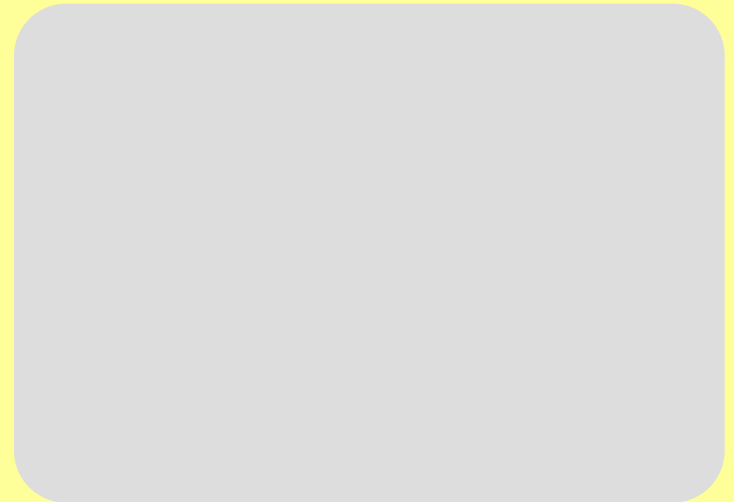


```
girl.address
```



구조체 변수의 초기화

```
struct name_card {  
    char    name[20];  
    struct phone_info {  
        int reg, first, second;  
    } phone;  
    char    address[60];  
    struct birth_info {  
        int year, month, date;  
    } birthday;  
} name_card;
```



구조체 포인터

```
struct name_card girl = {  
    "전지현"  
    { 11, 333, 7777 },  
    "모현면 왕산리"  
    { 1977, 7, 7 }  
};  
struct name_card *p;  
p = &girl;  
  
printf(" %s \n", girl.name);  
printf(" %s \n", p->name);
```