

CHAPTER 11. 구조체

- ✓ 구조체의 기본 개념과 특징을 이해한다.
- ✓ 구조체의 정의와 변수 선언 방법을 배운다.
- ✓ 구조체 배열을 배운다.
- ✓ 구조체 포인터를 배운다.
- ✓ 함수 간 구조체 전달에 대해 배운다.
- ✓ 동적 기억장소 할당을 이해하고 이용하는 방법을 배운다.
- ✓ char형 포인터와 동적 할당을 이용한 문자열 처리를 배운다.

구조체(Struct)

- 사용자 정의 자료형(user-defined data type)
- 다양한 자료형의 여러 값을 하나의 단위로 묶어서 관리

그림 11-1 가방과 구조체



(a) 여행 물품을 담은 가방

이름	학교	나이	평점
"홍길동"	"대한대학교"	20	4.2

구조체를 이용하여 저장한
홍길동에 관련된 4개의 데이터

(b) 문자열, 정수, 실수를 함께 저장한 구조체

○ 변수, 배열, 구조체 비교

- 한 개의 값 저장, 같은 종류의 여러 값 저장, 다른 종류의 여러 값 저장

그림 11-2 변수, 배열, 구조체 예

```
int sum = 0;
```

0

```
int array[5] = {9, 7, 6, 8, 5};
```

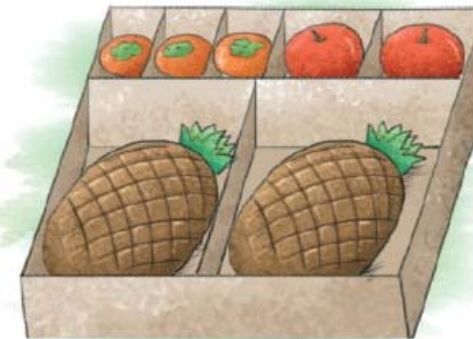
9	7	6	8	5
---	---	---	---	---

```
struct info freshman =
```

```
    {"홍길동", "대한대학교", 20, 4.2};
```

"홍길동"	"대한대학교"	20	4.2
-------	---------	----	-----

(a) 프로그램의 예



(b) 실생활의 예

구조체의 장점

- 서로 관련된 정보를 하나의 단위로 관리 가능
- 구조체 통째로 하나의 인수로서 함수로 전달 가능
- 한 개의 대입문으로 구조체 안의 모든 정보를 대입 가능: [그림 11-3]

그림 11-3 구조체 변수 간의 대입 예



구조체 정의

- C에서 제공하는 기본 자료형(int, double, char와 같은)이 아니므로 구조체 정의(템플릿 정의)를 먼저해야 구조체 변수를 선언할 수 있다.
- 형식

```
struct    구조체태그명
{
    자료형    멤버명1;
    자료형    멤버명2;
    ...
    자료형    멤버명n;
};
```

```
// 학생의 정보를 저장할 구조체
struct student_info
{
    char s_no[10];    // 학번
    char name[10];    // 이름
    int grade;        // 학년
    double GPA;       // 평점
};
```

주의

- 함수 외부에서 구조체 정의 → 정의 아래쪽의 모든 함수에서 사용 가능
함수 안에서 정의 → 이 함수에서만 사용 가능

예)

```
// 전자 제품의 정보를 저장할 구조체
struct product
{
    char SN[10];          // 제품 시리얼 번호
    int price;            // 제품 가격
    int sales[4];         // 분기별 판매수
};
```

```
// 3차원 공간의 한 점을 저장할 구조체
struct coordinate
{
    int x;                // x좌표
    int y;                // y좌표
    int z;                // z좌표
};
```

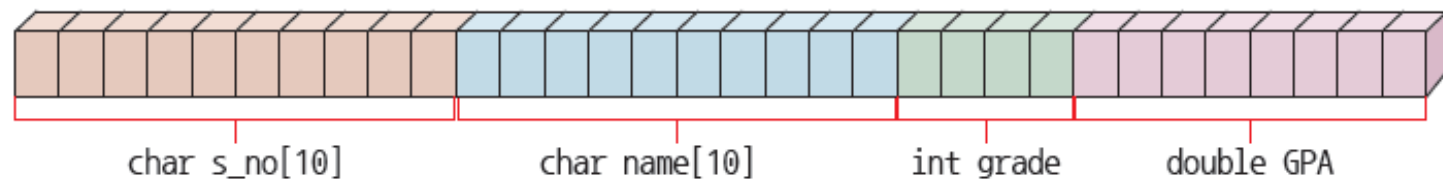
o 구조체 변수 선언

- 실제로 구조체 변수의 기억장소를 할당 받기 위해 필요
- 형식) `struct 구조체태그명 구조체변수명;`
- 예)

```
// 학생의 정보를 저장할 구조체
struct student_info
{
    char s_no[10];           // 학번
    char name[10];          // 이름
    int grade;               // 학년
    double GPA;              // 평점
};
```

```
struct student_info student;
```

struct student_info student;에 의해 할당된 student 변수의 기억장소 구조



○ 구조체 멤버 참조 연산자 .

구조체 멤버 참조 시 어떤 구조체 변수의 멤버인지 명시해야 함

- 형식

구조체변수명 . 멤버명

- 예

- student.GPA = 4.1;
- c_language.year = 2012;
- TV.price = 1200000;
- point.x = 10;
- scanf("%d", &point.x);

◦ 그림 11-8 구조체 사용 과정과 방법

1. 구조체 정의

예약어 구조체 태그명(사용자가 지정)

```
struct student_info  
{  
    char s_no[10]; // 학번  
    char name[10]; // 이름  
    int grade;     // 학년  
    double GPA;   // 평점  
};
```

구조체 멤버 선언

2. 구조체 변수 선언

```
struct student_info student;
```

3. 구조체 멤버 참조

```
student.grade = 4;
```

Program 11-1 1~3 라운드 게임 성적의 평균 구하기

p.528

```
4 struct game // 새로운 자료형 구조체 game 정의
5 {
6     char name[7]; // 선수 이름 저장 배열(한글 세 자 저장 가능)
7     int R1, R2, R3; // 각 라운드의 점수
8 };
9
10 int main()
11 {
12     struct game player; // 구조체 변수 선언
13     double avg;
14
15     printf("선수의 이름은? "); // 선수의 경기 정보 입력
16     scanf("%s", player.name);
17     printf("1, 2, 3라운드 점수는? ");
18     scanf("%d %d %d", &player.R1, &player.R2, &player.R3);
19
20
21     // 1, 2, 3라운드의 평균 점수 구하기
22     avg = (double) (player.R1 + player.R2 + player.R3) / 3;
23
24     // 게임 결과 출력하기
25     printf("%s선수의 게임 성적 평균 %.1lf점 \n", player.name, avg);
26
27     return 0;
28 }
```

실행결과

선수의 이름은? 홍길동
1, 2, 3라운드 점수는? 865 872 869
홍길동선수의 게임 성적 평균 868.7점

o 구조체 변수를 선언하면서 초기화하기

- 형식)

`struct` 구조체태그명 구조체변수명 = {멤버값1, ..., 멤버값n};

- 예)

- `struct student_info student = {"21121103", "홍길동", 4, 4.3};`
- `struct product TV = {"TV1209", 1200000, {250, 300, 290, 350}};`
- `struct coordinate point = {10, 20, 5};`

```
struct product
{
    char SN[10]; // 제품 시리얼 번호
    int price;   // 제품 가격
    int sales[4]; // 분기별 판매수
};
```

```
struct coordinate
{
    int x;      // x좌표
    int y;      // y좌표
    int z;      // z좌표
};
```

o 구조체 변수 간의 대입

- 같은 구조체형 변수끼리만 가능

- 형식)

구조체변수명1 = 구조체변수명2;

모든 멤버간 대입이 자동으로 처리됨

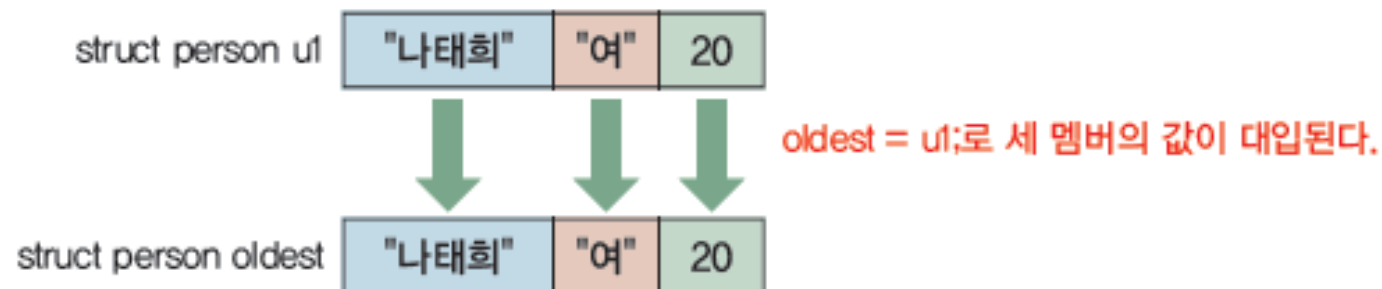
- 예)

```
struct coordinate point1 = {10, 20, 30}, point2;
```

```
point2 = point1;
```

point2.x = point1.x;
point2.y = point1.y;
point2.z = point1.z;

그림 11-9 구조체의 대입



○ 자료형 이름의 재정의

- 장점

- 프로그램의 이식성 증가
- 자료형 이름을 짧게 사용 → 가독성 증가

- 형식)

`typedef` 기존자료형이름 새자료형이름;

- 예)

- `typedef double REAL;`
`REAL average;` → 결국 `double average;`
- `typedef struct person PERSON;`
`PERSON user;` → 결국 `struct person user;`

○ 문제

- 자료: 물품을 구입한 세 명의 사용자 정보(이름, 성별, 나이)
- 출력: 나이가 가장 많은 사용자 정보

○ 분석

- 사용자의 이름, 성별, 나이를 하나로 묶어서 관리 → person 구조체 정의
- 최고령 사용자 정보 → person 구조체 변수에 저장
- 사용자 u1, u2 중 고령자를 oldest에 저장하기

```
if (u1.age < u2.age)
    oldest = u2;
```

나이 멤버를 비교해야 하므로

모든 멤버를 한꺼번에 대입하기 위해

실행결과		
이름	성별	나이
나태희	여	20
유현빈	남	29
나원빈	남	25
〈 최고령 사용자 〉		
유현빈	남	29

Program 11-2 세 명의 사용자 중 최고령자 정보 출력하기

p.534

```
1  #include <stdio.h>
2
3  // 구조체 정의
4  struct person
5  {
6      char name[7];      // 이름
7      char gender[3];    // 성별
8      int age;           // 나이
9  };
10
11 int main()
12 {
13     // 세 명의 사용자 정보를 구조체 변수에 저장하기
14     struct person u1 = {"나태희", "여", 20}, u2 = {"유현빈", "남", 29},
15                     u3 = {"나원빈", "남", 25};
16
17     struct person oldest; // 최고령자의 정보를 저장할 구조체 변수
```

u1	"나태희"	"여"	20
u2	"유현빈"	"남"	29
u3	"나원빈"	"남"	25

Program 11-2 세 명의 사용자 중 최고령자 정보 출력하기

p.534

```
19  if (u1. age > u2. age)
20  {
21      // u1과 u3 중 고령자 찾기
22      if (u1. age > u3. age)
23          oldest = u1;
24      else
25          oldest = u3;
26  }
27  else
28  {
29      // u2와 u3 중 고령자 찾기
30      if (u2. age > u3. age)
31          oldest = u2;
32      else
33          oldest = u3;
34  }
```

구조체의 장점: 세 멤버의 대입을 한 문장으로 해결

u1	"나태희"	"여"	20
u2	"유현빈"	"남"	29
u3	"나원빈"	"남"	25

oldest			
--------	--	--	--

Program 11-2 세 명의 사용자 중 최고령자 정보 출력하기

p.534

```

36 printf(" 이름 성별 나이 \n");
37 printf("=====\n");
38 printf("%s\t %s\t %2d\n", u1.name, u1.gender, u1.age);
39 printf("%s\t %s\t %2d\n", u2.name, u2.gender, u2.age);
40 printf("%s\t %s\t %2d\n", u3.name, u3.gender, u3.age);
41 printf("=====\n\n");
42 printf(" << 최고령 사용자 >>\n\n");
43 printf(" %s\t %s\t %d\n\n", oldest.name, oldest.gender, oldest.age);
44
45 return 0;
46 }

```

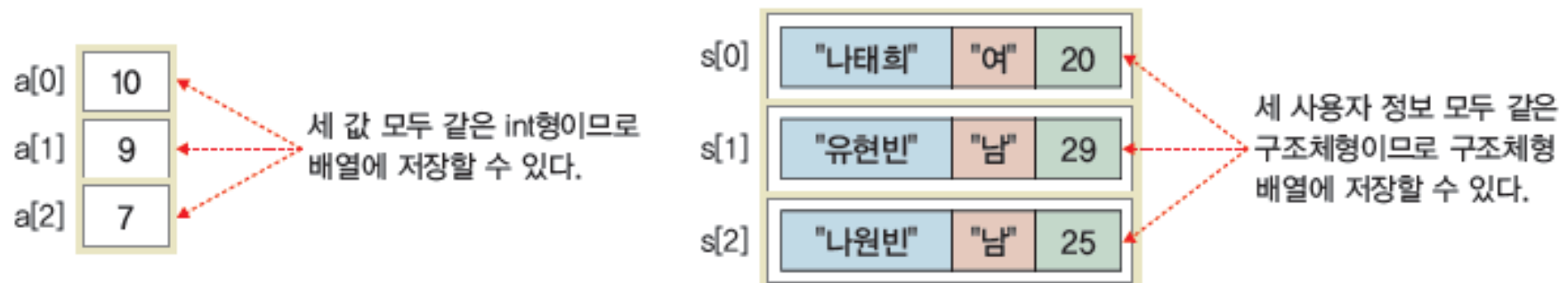
u1	"나태희"	"여"	20
u2	"유현빈"	"남"	29
u3	"나원빈"	"남"	25
oldest	"유현빈"	"남"	29

실행결과			
이름 성별 나이			
나태희	여	20	
유현빈	남	29	
나원빈	남	25	
<< 최고령 사용자 >>			
유현빈	남	29	

Q) 사용자가 100명이라면?
→ 구조체 배열로 해결

- 종류가 다른 3000개의 정보를 원소수가 3000개인 배열에 저장할 수 있을까?
 - 같은 구조체 형 정보 1000개는 원소수가 1000개인 배열에 저장할 수 있을까?
- ➔ 구조체도 일반 자료형처럼 배열의 자료형으로 사용 가능

그림 11-11 int형 배열과 구조체형 배열



o 구조체 배열 선언

동일 구조체형 자료를 구조체형 배열에 저장 가능

▪ 형식)

`struct` 구조체태그명 구조체배열명[원소수];

7장의 배열 선언 참조
자료형 배열명[원소수];

▪ 예)

- `struct person` user[100]; // 사용자 100명 구조체 정보 저장 배열
- `struct student_info` student[200];
- `struct product` smart[10];

o 구조체 배열의 선언과 동시에 초기화

▪ 형식)

```
struct 구조체태그명 구조체배열명[원소수] = {  
    {멤버 값1, 멤버 값2, ..., 멤버 값n},  
    {멤버 값1, 멤버 값2, ..., 멤버 값n},  
    ...,  
    {멤버 값1, 멤버 값2, ..., 멤버 값n} };
```

배열

구조체

▪ 예)

```
struct person user[100] = { {"나태희", "여", 20}, {"유현빈", "남", 29},  
    {"나원빈", "남", 25}, ..., {"이성룡", "남", 45} };
```

```
struct person  
{  
    char name[7];  
    char gender[3];  
    int age;  
};
```


구조체 배열의 원소 참조

구조체배열명[첨자]

0 ~ (원소수-1)

구조체 배열 원소의 멤버 참조

구조체배열명[첨자].멤버명

멤버 참조 연산자

구조체 자료

예)

- `user[0].age = 20;` // 첫 번째 사용자의 나이
- `user[99].age = 32;` // 백 번째 사용자의 나이
- `smart[0].sales[2] = 290;` // 첫 번째 제품의 3사분기 판매량

1. 구조체 템플릿 구상

2. 구조체 템플릿 정의

이름	성별	나이
----	----	----

```
struct person
{
    char name[7];
    char gender[3];
    int age;
};
```

3. 구조체 배열 선언과 초기화

```
struct user[10] = { {"나태희", "여", 20},
                   {"유현빈", "남", 29}, {"나원빈", "남", 25}, ...,
                   {"김해수", "여", 35} };
```

4. 배열 원소, 멤버 참조

배열 원소 참조

user[0]	"나태희"	"여"	20
user[1]	"유현빈"	"남"	29
user[2]	"나원빈"	"남"	25
⋮	⋮	⋮	⋮
user[9]	"김해수"	"여"	35

멤버 참조

user[9].age

```

1  #include <stdio.h>
2  #include <string.h> // strcmp 함수를 위한 헤더 파일
3  #define N 10        // 사용자 수를 매크로 상수로 정의
4
5
6  struct person       // 구조체 정의
7  {
8      char name[7];    // 이름
9      char gender[3];  // 성별
10     int age;         // 나이
11 };
12
13 int main()
14 {
15     // N개의 구조체형 원소를 갖는 배열의 선언과 초기화
16     struct person user[N] = {{"나태희", "여", 20}, {"유현빈", "남", 29},
17                               {"나원빈", "남", 25}, {"문건영", "여", 22}, {"소지법", "남", 25},
18                               {"나보배", "여", 29}, {"장도건", "남", 32}, {"고수영", "여", 29},
19                               {"이나라", "여", 31}, {"김해수", "여", 35} };


```

```

20
21  int male = 0, female = 0, sum = 0, i; // 남자수, 여자수
22
23  // 남녀 수와 전체 나이의 합 구하기
24  for (i=0; i<N; i++)
25  {
26      // user[i]의 gender 멤버 성별에 따라 남자수 또는 여자수를 1 증가
27      if ( strcmp(user[i].gender, "남") == 0)
28          male++;
29      else
30          female++;
31
32      sum += user[i].age; // 사용자의 나이를 sum에 누적
33  }

```

두 문자열이 같다면
0이 반환됨



```

35  // 결과 출력하기
36  printf(" 이름 성별 나이 \n");
37  printf("=====\n");
38  for (i=0; i<N; i++)
39      printf("%s\t %s\t %d\n", user[i].name, user[i].gender, user[i].age);
40  printf("=====\n\n");
41  printf(">> 남자: %d명\n", male);
42  printf(">> 여자: %d명\n", female);
43  printf(">> 평균 연령: %d살\n", sum / N);
44
45  return 0;
46  }

```

. 아래값 제거

실행결과

이름	성별	나이
나태희	여	20
유현빈	남	29
나원빈	남	25
문건영	여	22
소지법	남	25
나보배	여	29
장도건	남	32
고수영	여	29
이나라	여	31
김해수	여	35

>> 남자: 4명
 >> 여자: 6명
 >> 평균 연령: 27살

실습

○ [프로그램 11-2]와 [프로그램11-3]을 참고

- [프로그램 11-3]에서 앞의 5명만 구조체 배열에 저장하고
- 5명의 정보를 출력한 후
- 나이가 제일 많은 사람의 정보를 max에 저장하여
- max의 멤버를 이용하여 최고령자 정보를 출력하기

나태희 20세
유현빈 29세
나원빈 25세
문건영 22세
소지법 25세

user[o]의 name과 age 멤버값

최고령자: 유현빈(남) 29세

max의 name, gender, age 멤버값

11.6.1 구조체 배열을 이용한 직원의 평균 급여 구하기

o 구조체 배열(프로그램 11-6) vs. 일반 배열(프로그램 11-7)

- 두 프로그램을 통해 구조체의 장점 이해

o 구조체 배열 이용: 프로그램 11-6

- 문제
 - 입력: 직원 5명의 이름, 아이디, 급여 정보
 - 출력: 급여가 평균 이상인 직원의 아이디와 급여 정보 출력

11.6.1 구조체 배열을 이용한 직원의 평균 급여 구하기

- 분석

- 한 직원의 관련 정보 이름, 아이디, 급여를 구조체에 저장

employee_info 구조체의 템플릿

char name[7]	char id[10]	int salary
--------------	-------------	------------

- 직원 5명의 정보는 구조체 배열에 저장

struct employee_info employee[5];

employee[0]	"나태희"	"natae"	5000
employee[1]	"유현빈"	"wiseguy"	4000
employee[2]	"나원빈"	"circle"	6000
employee[3]	"문건영"	"young"	5500
employee[4]	"소지법"	"cow"	4500

```

1  #include <stdio.h>
2  #include <string.h>      // strcpy 함수를 위한 헤더 파일
3  #define SIZE 5          // 인원수
4
5  int main()
6  {
7      char name[SIZE][7]; // SIZE개의 이름 문자열을 저장할 2차원 배열
8      char id[SIZE][10];  // SIZE개의 아이디 문자열을 저장할 2차원 배열
9      int salary[SIZE];    // SIZE개의 급여를 저장할 1차원 배열
10     int average= 0, sum= 0, i, temp_int, pass; // 급여 평균, 급여 합계
11     char temp_string[10]; // 문자열을 임시로 저장할 곳

```

name[0]		id[0]		salary[0]	
name[1]		id[1]		salary[1]	
name[2]		id[2]		salary[2]	
name[3]		id[3]		salary[3]	
name[4]		id[4]		salary[4]	

Program 11-7 구조체를 이용하지 않고 직원 정보를 급여순으로 정렬하기

p.551

```

13 // 직원 5명의 정보 입력 및 급여 누적
14 printf(">> 직원의 정보(이름, ID, 급여)를 입력하세요. << \n");
15 for (i=0; i<SIZE; i++)
16 {
17     printf("%d번: ", i+1);
18     scanf("%s %s %d", name[i], id[i], &salary[i]);
19     sum += salary[i]; // 급여 합계
20 }
21 average = sum / SIZE; // 급여 평균
22
23 // 급여가 평균 이상인 직원의 id와 급여 출력
24 printf("\n-----\n");
25 printf("급여가 %d만원(평균) 이상인 직원 정보 ", average);
26 printf("\n-----\n");
27 for (i=0; i<SIZE; i++)
28 {
29     if (salary[i] >= average)
30         printf("\t%s\t %d만원 \n", id[i], salary[i]);
31 }
32 printf("-----\n");

```

name[0]	나태희
name[1]	유현빈
name[2]	나원빈
name[3]	문건영
name[4]	소지법

id[0]	natae
id[1]	wiseguy
id[2]	circle
id[3]	young
id[4]	cow

salary[0]	5000
salary[1]	4000
salary[2]	6000
salary[3]	5500
salary[4]	4500

Program 11-7 구조체를 이용하지 않고 직원 정보를 급여순으로 정렬하기

p.551

```

34  // 급여의 오름차순으로 직원의 정보를 정렬하기(버블 정렬)
35  for (pass=1; pass<SIZE; pass++)
36      for (i=0; i<SIZE-pass; i++)
37      {
38          // 앞 직원의 급여가 더 크다면 서로 교환하기
39          if (salary[i] > salary[i+1])
40          {
41              // 이름 교환
42              strcpy(temp_string, name[i]);
43              strcpy(name[i], name[i+1]);
44              strcpy(name[i+1], temp_string);
45              // 아이디 교환
46              strcpy(temp_string, id[i]);
47              strcpy(id[i], id[i+1]);
48              strcpy(id[i+1], temp_string);
49              // 급여 교환
50              temp_int = salary[i];
51              salary[i] = salary[i+1];
52              salary[i+1] = temp_int;
53          }
54      }

```

구조체를 이용 시
구조체 배열 원소 교환
3문장이면 됨

name[0]	나태희
name[1]	유현빈
name[2]	나원빈
name[3]	문건영
name[4]	소지법

id[0]	natae
id[1]	wiseguy
id[2]	circle
id[3]	young
id[4]	cow

salary[0]	5000
salary[1]	4000
salary[2]	6000
salary[3]	5500
salary[4]	4500

Program 11-7 구조체를 이용하지 않고 직원 정보를 급여순으로 정렬하기

p.551

```
56 // 정렬 결과 출력하기
57 printf("\n-----\n");
58 printf("\t급여순 정렬 결과 ");
59 printf("\n-----\n");
60 for (i=0; i<SIZE; i++)
61     printf("\t%s\t %d만원 \n", id[i], salary[i]);
62 printf("\n-----\n");
63
64 return 0;
65 }
```

실행결과

>> 직원의 정보(이름, ID, 급여)를 입력하세요. <<

1번: 나태희 natae 5000
2번: 유현빈 wiseguy 4000
3번: 나원빈 circle 6000
4번: 문건영 young 5500
5번: 소지법 cow 4500

급여가 5000만원(평균) 이상인 직원 정보

natae	5000만원
circle	6000만원
young	5500만원

급여순 정렬 결과

wiseguy	4000만원
cow	4500만원
natae	5000만원
young	5500만원
circle	6000만원

Program 11-6 직원의 평균 급여와 평균 급여 이상 직원의 정보 출력 p.550

```
1  #include <stdio.h>
2  #define SIZE 5          // 인원수
3
4  struct employee_info    // 구조체 정의
5  {
6      char name[7];       // 최대 6개 문자(한글 세 개)를 저장할 배열 멤버
7      char id[10];        // 최대 9개 문자를 저장할 배열 멤버
8      int salary;         // 급여를 저장할 정수형 멤버
9  };
10
11 int main()
12 {
13     struct employee_info employee[SIZE]; // 구조체 배열 선언
14     int average = 0, sum = 0, i;         // 급여 평균, 급여 합계
```

Program 11-6 직원의 평균 급여와 평균 급여 이상 직원의 정보 출력 p.550

```
16 // 직원 5명의 정보 입력 및 급여 누적
17 printf(">> 직원의 정보(이름, ID, 급여)를 입력하세요. << \n");
18 for (i=0; i<SIZE; i++)
19 {
20     printf("%d번: ", i+1);
21     scanf("%s %s %d",
22         empl oyee[i].name, empl oyee[i].id, &empl oyee[i].sal ary);
23     sum += empl oyee[i].sal ary; // 급여 합계 계산
24 }
25 average = sum / SIZE; // 급여 평균(원단위) 계산
```

실행결과

```
>> 직원의 정보(이름, ID, 급여)를 입력하세요. <<
1번: 나태희 natae 5000
2번: 유현빈 wiseguy 4000
3번: 나원빈 circle 6000
4번: 문건영 young 5500
5번: 소지법 cow 4500
```

Program 11-6 직원의 평균 급여와 평균 급여 이상 직원의 정보 출력 p.550

```
27 // 급여가 평균 이상인 직원의 id와 급여 출력
28 printf("\n-----\n");
29 printf(" 급여가 %d만원(평균) 이상인 직원 정보 ", average);
30 printf("\n-----\n");
31 for (i=0; i<SIZE; i++)
32 {
33     if (employee[i].salary >= average)
34         printf("\t%s\t %d만원 \n",
35                employee[i].id, employee[i].salary);
36 }
37 printf("-----\n");
38 return 0;
39 }
```

급여가 5000만원(평균) 이상인 직원 정보

natae	5000만원
circle	6000만원
young	5500만원

○ [프로그램 11-7]

- [프로그램 11-6]을 구조체를 이용하지 않도록 수정하고 급여에 대해 오름차순으로 정렬한 결과를 출력

○ 분석

- 이름과 아이디
 - SIZE개의 문자열을 저장하는 char형 2차원 배열을 이용
- 급여
 - SIZE개의 정수를 저장하도록 int형 1차원 배열을 이용
- 급여 오름차순으로 정렬하기
 - 정렬과정에서 두 급여의 위치 교환 시 해당 이름과 아이디도 교환해야 함
 - 교환시 세 문장이 필요 → 이름,아이디, 급여 교환 시 9문장이 필요
비교) 구조체를 이용한 경우 구조체 단위로 교환하므로 3문장만 필요

name[0]	나태희
name[1]	유현빈
name[2]	나원빈
name[3]	문건영
name[4]	소지법

id[0]	natae
id[1]	wiseguy
id[2]	circle
id[3]	young
id[4]	cow

salary[0]	5000
salary[1]	4000
salary[2]	6000
salary[3]	5500
salary[4]	4500

구조체 포인터

연결 리스트(linked list), 트리(tree)와 같은 자료구조 구현에 유용

구조체 포인터 변수 선언

```
struct 구조체태그명 * 구조체포인터변수명;
```

포인터가 구조체를 가리키기

```
구조체포인터변수명 = &구조체변수명;
```

- '=' 좌우의 변수는 같은 구조체형

예)

```
struct person shopper;
```

```
struct person *ptr;
```

```
ptr = &shopper;
```



포인터를 이용한 구조체 멤버 참조: * 이용하기

(*구조체포인터변수명) . 멤버

구조체 포인터 변수가
가리키는 구조체

- 예)
 - (*ptr).age = 20;
 - printf("%d", (*ptr).age);



- (주) *ptr.age → 에러
→ *보다 .이 우선순위가 높으므로 *(ptr.age)로 해석됨

○ 포인터를 이용한 구조체 멤버 참조: -> 이용하기

구조체포인터변수명 -> 멤버명

구조체 포인터 변수가
가리키는 구조체의

(*구조체포인터변수명) . 멤버명

■ 예)

- printf("%s", ptr->age);
- ptr->age = 30;



- (*ptr).age에 비해 의미적(가리키는 →)으로나 타이핑으로 더 편리함

o [프로그램 11-4]

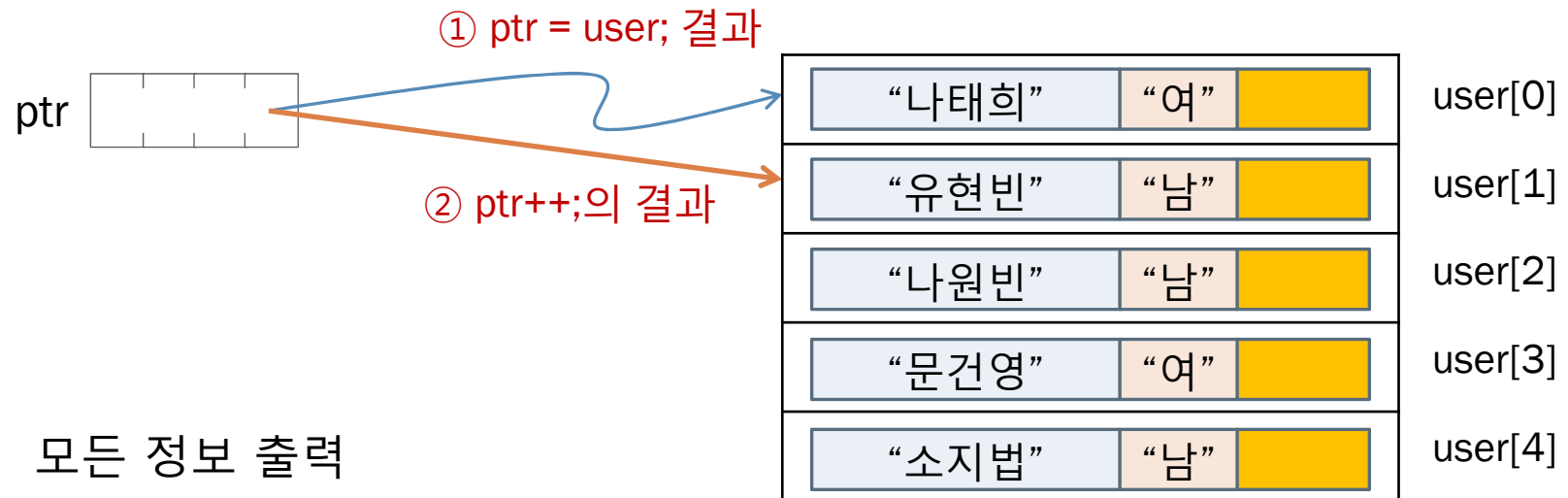
- 문제
 - 자료: 사용자 5명의 이름과 성별
 - 입력: 사용자 5명의 나이
 - 출력: 사용자의 모든 정보 출력
- 각 사용자의 정보 이름, 성별, 나이는 구조체에 저장
5명의 정보는 구조체 배열에 저장

“나태희”	“여”		user[0]
“유현빈”	“남”		user[1]
“나원빈”	“남”		user[2]
“문건영”	“여”		user[3]
“소지법”	“남”		user[4]

■ 나이 입력

① ptr = user;

② ptr++;로 ptr이 다음 원소 가리키게 하면서 배열 원소의 나이 멤버를 입력



■ 모든 정보 출력

- ptr = user;

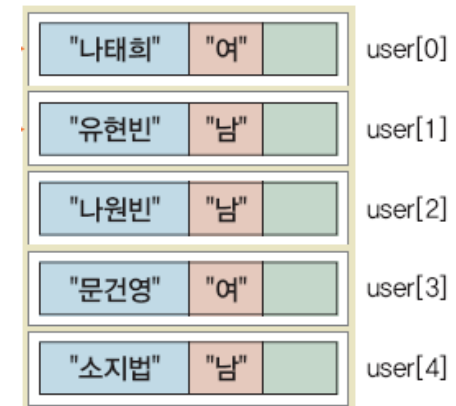
- ptr++;로

ptr이 다음 원소 가리키게 하면서 배열 원소의 모든 멤버를 출력

Program 11-4 구조체 포인터를 이용 구조체 배열 출력하기

p.543

```
1  #include <stdio.h>
2  #define N 5
3
4  struct person                // 구조체 정의
5  {
6      char name[7], gender[3]; // 이름, 성별
7      int age;                 // 나이
8  };
9
10 int main()
11 {
12     int i;
13     struct person *ptr;       // 구조체 포인터 선언
14
15     // N개의 구조체 원소를 갖는 배열 선언과 초기화
16     struct person user[N] = { {"나태희", "여"}, {"유현빈", "남"},
17                                {"나원빈", "남"}, {"문건영", "여"}, {"소지법", "남"} };
18
```

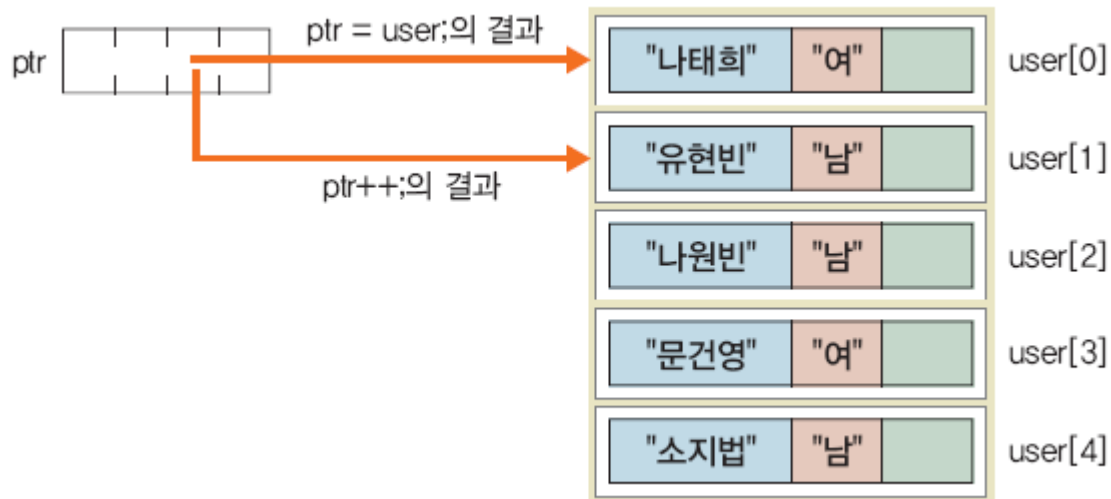


나이는 직접 입력받으므로
초기화하지 않음
→ 끝부분의 멤버들은
초기화 생략 가능

Program 11-4 구조체 포인터를 이용 구조체 배열 출력하기

p.543

```
19 // ptr 포인터가 차례대로 다음 원소를 가리키게 한 후 사용자의 나이를 입력받기
20 ptr = user; // ptr이 user 배열을 가리키게 하기
21 for (i=0; i<N; i++)
22 {
23     // user[i]의 age 멤버의 값을 입력받기
24     printf("%s의 나이는? ", (*ptr).name);
25     scanf("%d", &(*ptr).age);
27     ptr++; // ptr이 user 배열의 다음 원소를 가리키게 함
28 }
```



실행결과

나태희의 나이는? 20
유현빈의 나이는? 29
나원빈의 나이는? 25
문건영의 나이는? 22
소지법의 나이는? 25

이름	성별	나이
나태희	여	20
유현빈	남	29
나원빈	남	25
문건영	여	22
소지법	남	25

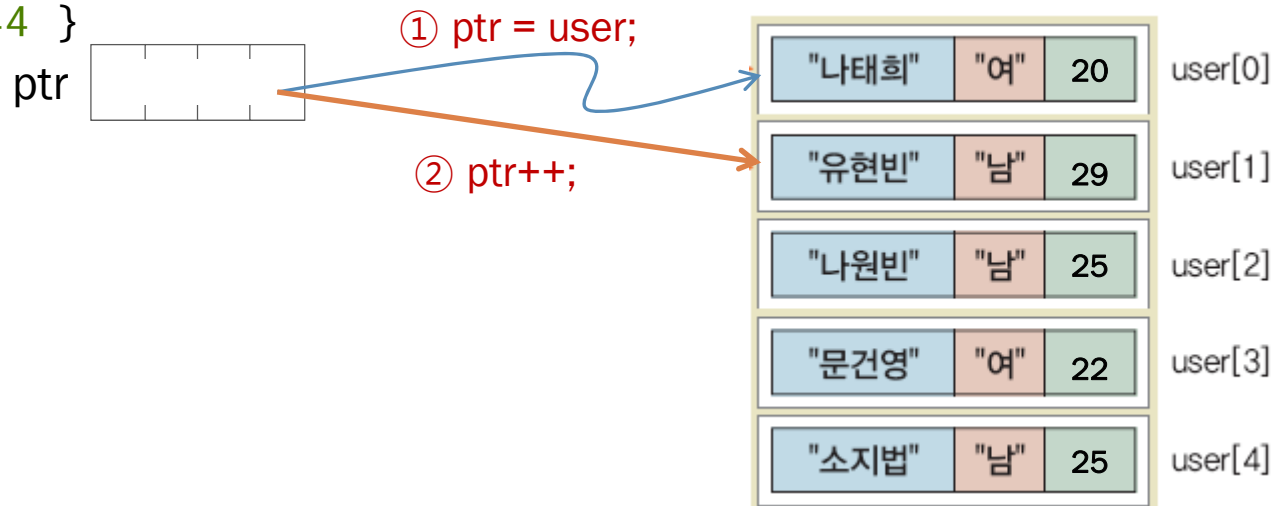
Program 11-4 구조체 포인터를 이용 구조체 배열 출력하기

p.543

```

31  printf("\n 이름 성별 나이 \n");
32  printf("=====\n");
33  ptr = user; // ptr을 다시 첫 원소를 가리키게 하기
34  for (i=0; i<N; i++)
35  {
36      // ptr을 이용 배열 원소의 멤버 출력
37      printf("%s\t %s\t %d\n", ptr->name, ptr->gender, ptr->age);
39      ptr++; // ptr이 user 배열의 다음 원소를 가리키게 함
40  }
41  printf("=====\n\n");
43  return 0;
44 }

```



실행결과

나태희의 나이는? 20
 유현빈의 나이는? 29
 나원빈의 나이는? 25
 문건영의 나이는? 22
 소지법의 나이는? 25

이름	성별	나이
나태희	여	20
유현빈	남	29
나원빈	남	25
문건영	여	22
소지법	남	25

11.5.1 값에 의한 호출 p.545

- 값에 의한 호출 방식을 이용한 구조체 전달
 - 함수가 호출되면 매개변수는 구조체에 해당하는 기억 공간을 할당받으며, 인수의 각 멤버의 값이 매개변수의 멤버의 값으로 저장된다.

함수 호출

함수명 (구조체인수명);

동일 구조체형

함수 정의

```
반환값형    함수명(struct 구조체태그명 매개변수명)
{
    :
}
```

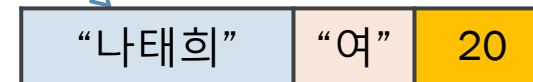
```
int main()
{
    :
    print(buyer);
    :
}
```

```
void print(struct person shopper)
{
    :
}
```

buyer



shopper



구조체의 값에 의한 호출 방식의 문제

- 인수와 동일한 기억공간이 필요
 - 각 멤버 값이 복사되어 저장됨
- 구조체 크기가 클수록 주기억장치의 낭비와 실행 시간이 증가됨

○ 주소에 의한 호출 방식을 이용한 구조체 전달

- 호출된 함수에서는 구조체 포인터를 이용 자신을 호출한 함수의 구조체를 간접적으로 참조 가능
- 구조체 크기와 상관없이 매개변수인 포인터의 크기는 4바이트
- 멤버 값 복사가 필요 없음

함수 호출

함수명 (&구조체인수명);

구조체의 주소가 전달됨

함수 정의

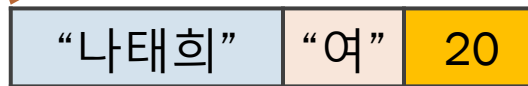
```
반환값형    함수명(struct 구조체태그명 *매개변수명)
{
    : 매개변수->멤버로 호출한 함수의 멤버
참조 가능
}
```

11.5.2 주소에 의한 호출

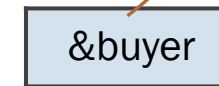
```
int main()
{
    :
    print(&buyer);
    :
}
```

```
void print(struct person *ptr)
{
    :
}
```

buyer



ptr



print 함수에서는
buyer->age 또는 **(*buyer).age**로
main의 구조체 멤버 참조 가능

11.5.2 주소에 의한 호출

o [프로그램 11-5]

[프로그램 11-4]를
구조체의 입력은 주소에 의한 호출을,
구조체의 출력은 값에 의한 호출을 이용하도록 수정
→ 두 방법을 비교

Program 11-5 구조체를 함수로 전달하여 입출력하기

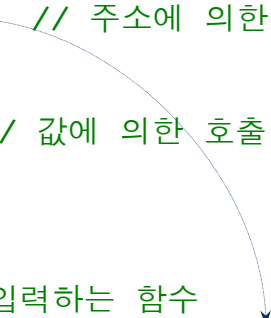
p.546

```
1  #include <stdio.h>
2  #define N 5
3
4  struct person                // 구조체 정의
5  {
6      char name[7], gender[3]; // 이름, 성별
7      int age;                 // 나이
8  };
9
11 void input_by_address(struct person *ptr); // 주소에 의한 호출
12 void output_by_value(struct person shopper); // 값에 의한 호출
13
14 int main()
15 {
16     struct person buyer;
17
18     input_by_address(&buyer); // 주소에 의한 호출
19
20     printf("\n>> 구매자 정보 : ");
21     output_by_value(buyer);    // 값에 의한 호출
22
23     return 0;
24 }
```

Program 11-5 구조체를 함수로 전달하여 입출력하기

p.546

```
14 int main()
15 {
18     input_by_address(&buyer); // 주소에 의한 호출 방식
19     :
21     output_by_value(buyer); // 값에 의한 호출 방식
24 }
25
26 // ptr이 가리키는 곳의 구조체에 자료를 입력하는 함수
27 void input_by_address(struct person * ptr)
28 {
29     printf("이름은? "); scanf("%s", ptr->name);
30     printf("성별은(남 또는 여)? "); scanf("%s", ptr->gender);
31     printf("나이는? "); scanf("%d", &ptr->age);
32 }
33
34 // 전달된 구조체의 멤버를 출력하는 함수
35 void output_by_value(struct person shopper)
36 {
37     printf("%s(%s) %d세\n", shopper.name, shopper.gender, shopper.age);
38 }
```



함수간 구조체 배열 전달

- 주소에 의한 호출 방식

함수 호출

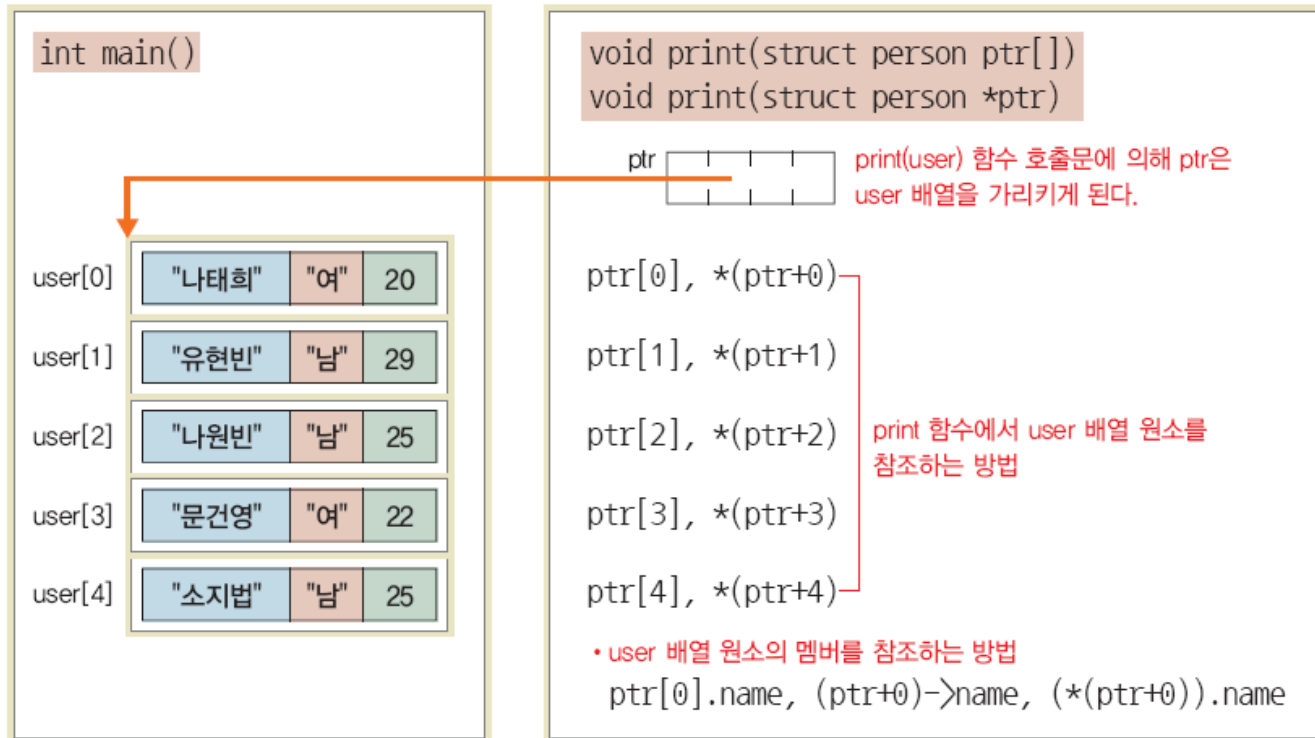
함수명 (구조체배열명);

또는 *배열명

함수 정의

```
반환값형   함수명(struct 구조체태그명 배열명[원소수])
{
    • 배열 원소(구조체) 참조
      배열명[첨자] 또는 *포인터명

    • 배열 원소(구조체)의 멤버 참조
      배열명[첨자]. 멤버명
      (*포인터명). 멤버명
      포인터명->멤버명
}
```



예) [그림 11-14]의 5명의 정보가 저장된 user 배열을 출력하는 함수 print

```

1 void print(struct person ptr[])
2 {
3     int i;
5     for (i=0; i<5; i++)
6     {
7         printf("%s\t %s\t %d\n",
8             ptr[i].name, ptr[i].gender, ptr[i].age);
9     }
10 }
```

문제

- 입력 자료가 있는 한 최대 100명까지 사용자 이름, 성별, 나이를 입력받은 후 사용자 정보를 출력하기
- 사용자 정보는 [프로그램 11-5]에서 정의한 person형 구조체에 저장

분석

- 최대 100명 정보를 저장할 person형 구조체 배열을 선언
- 실제 배열에 입력된 사용자 수
 - 전역 변수 count를 이용
 - input 함수에서 사용자 정보가 입력될 때마다 1씩 증가
 - output 함수에서는 count를 이용 실제 배열에 저장된 정보만큼 출력
- input 함수
 - 사용자 정보를 배열에 입력
- output 함수
 - 배열에 저장된 사용자 정보를 출력

```

2 #define SIZE 100    // 배열의 크기 지정
4 struct person      // 구조체 정의
5 {
6     char name[7], gender[3];    // 이름, 성별
7     int age;                    // 나이
8 };
11 void input(struct person ptr[]);
12 void print(struct person *ptr);
13
14 int count;
15
16 int main()
17 {
18     struct person user[SIZE]; // 원소가 SIZE개인 구조체 배열 user 선언
19
20     printf(" 최대 %d명의 사용자 정보를 입력받고 출력합니다. \n", SIZE);
21     input(user); // 사용자 정보를 배열에 입력하기
22     print(user); // 배열에 저장된 사용자 정보를 출력하기
23
24     return 0;
25 }

```

입력된 전체 사용자 수 저장 변수,
전역 변수,
자동으로 0으로 초기화 됨

```

27 void input(struct person ptr[]) // 배열에 사용자 정보 입력 함수
28 {
29     int reply;
30
31     do
32     { // user[i]의 각 멤버에 사용자 정보를 입력하기
33         printf("%2d. 이름은? ", count+1); scanf("%s", ptr[count].name);
34         printf(" 성별은(남 또는 여)? "); scanf("%s", ptr[count].gender);
35         printf(" 나이는? "); scanf("%d", &ptr[count].age);
36
37         count++;
38     }
39     while (count == SIZE) // 지금까지 입력한 사용자 수가 배열 원소수와 같다면 입력을 그만두기
40     {
41         printf("더 이상 사용자 정보를 입력할 공간이 없습니다. ");
42         break;
43     }
44     // 사용자 정보의 입력 여부를 묻기
45     printf(" 계속 입력하려면 1, 그만하려면 0을 입력하세요. ");
46     scanf("%d", &reply);
47 } while (reply != 0); // 그만하기를 원하지 않는다면 다시 반복하기
48 }

```

실제 배열에 입력된 사용자 수를 / 증가

Program 11-8 구조체 배열을 함수로 전달하여 사용자 정보를 입출력하기

p.555

```
53 // 배열을 포인터 변수로 전달받아 배열 내용을 출력하는 함수
54 void print(struct person *ptr)
55 {
56     int i;
57
58     printf("\n 이름 성별 나이 \n");
59     printf("=====\n");
60     for (i=0; i < count; i++)
61     {
62         printf("%s\t %s\t %d\n", (ptr+i)->name,
63             (ptr+i)->gender, (ptr+i)->age);
64     }
65     printf("=====\n");
66 }
```

*input 함수에서 구한
정보가 입력된 사용자 수*

실행결과

최대 100명의 사용자 정보를 입력받고 출력합니다.

1. 이름은? 나태희
성별은(남 또는 여)? 여
나이는? 20
계속 입력하려면 1, 그만하려면 0을 입력하세요. 1
2. 이름은? 유현빈
성별은(남 또는 여)? 남
나이는? 29
계속 입력하려면 1, 그만하려면 0을 입력하세요. 1
3. 이름은? 나원빈
성별은(남 또는 여)? 남
나이는? 25
계속 입력하려면 1, 그만하려면 0을 입력하세요. 0

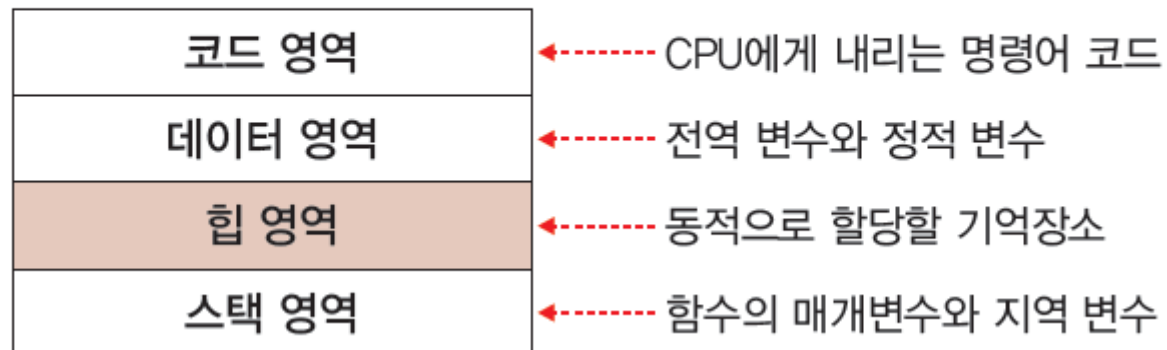
이름 성별 나이

나태희	여	20
유현빈	남	29
나원빈	남	25

11.7.1 주기억장치의 네 가지 영역

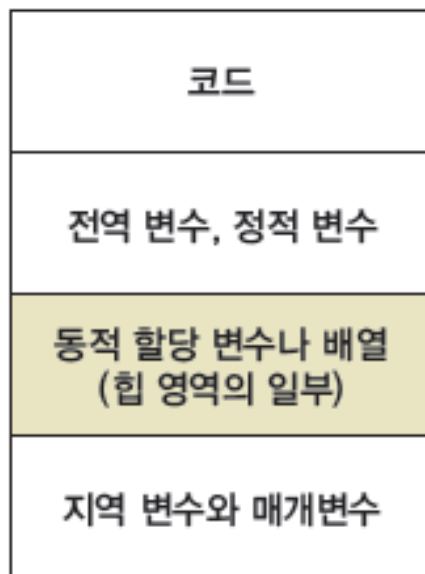
- 운영체제가 사용하는 영역을 제외하고 스택(stack), 힙(heap), 데이터, 코드 영역 네 가지로 구분
- 실행될 프로그램은 이 네 가지 영역의 일부에 적절히 저장(적재)

그림 11-15 주기억장치의 네 가지 영역



- 실행 프로그램의 코드, 데이터, 스택 영역의 크기
 - 프로그램이 실행될 때부터 결정되며 변하지 않음
- 동적 할당 영역의 크기
 - 프로그램 실행 동안 할당과 해제에 의해 계속 변함

그림 11-16 실행 프로그램의 구조



- 프로그램을 실행하는 동안 기억 공간을 요청하면 힙 영역의 일부를 동적으로 할당 받을 수 있다.
- 할당 받은 기억 공간을 더 이상 사용하지 않는다면 해제하여 다음 동적 할당이나 다른 프로그램에서 사용하게 할 수 있다.
- 프로그램이 실행되는 동안 할당과 해제에 의해 크기가 계속 변한다.

o 정적 기억장소 할당

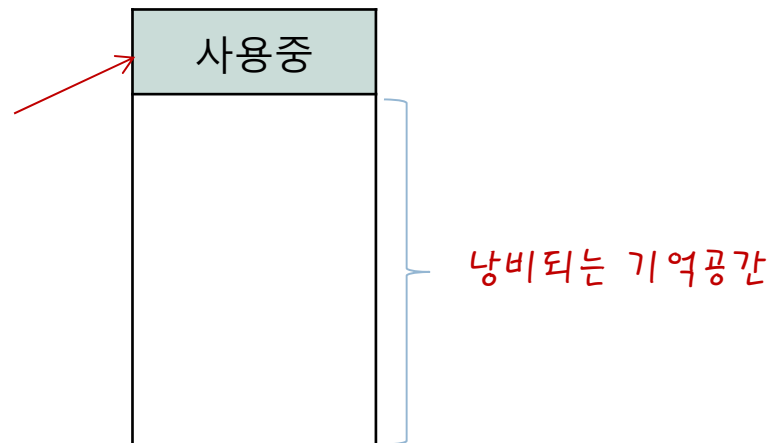
- 프로그램 컴파일 시 할당받을 기억장소의 크기가 결정
- 프로그램 실행 시작 전에 정적으로(statically) 할당됨
 - 고정된 크기의 기억장소를 할당바다 실행 끝까지 사용함
 - 실행이 끝나면 할당된 기억장소가 해제
- 전역 변수, 정적 변수가 정적 할당에 해당
- 지역 변수는 프로그램을 실행하는 동안 함수가 호출되면 스택 영역의 기억장소를 할당받지만 그 크기는 컴파일 시점에 이미 결정된 것으로 정적 할당에 해당
- 예
 - `int array[100];` → 4바이트(int형) * 100(원소수) = 400바이트
 - `double average;` → 8바이트(double형)
 - `double GPA[4][50];` → 8바이트(double형)*4(행수)*50(열수) = 1600바이트

정적 기억장소 할당의 문제점

- int score[100];
 - 학생이 100명이 넘는 경우에는 사용 불가
 - 이를 해결하기 위해 int score[1000];으로 한 경우
실제로 저장할 점수가 200개라면 나머지 800개의 기억 공간은 낭비

int score[1000];

배열에서 실제로 데이터가
저장된 부분

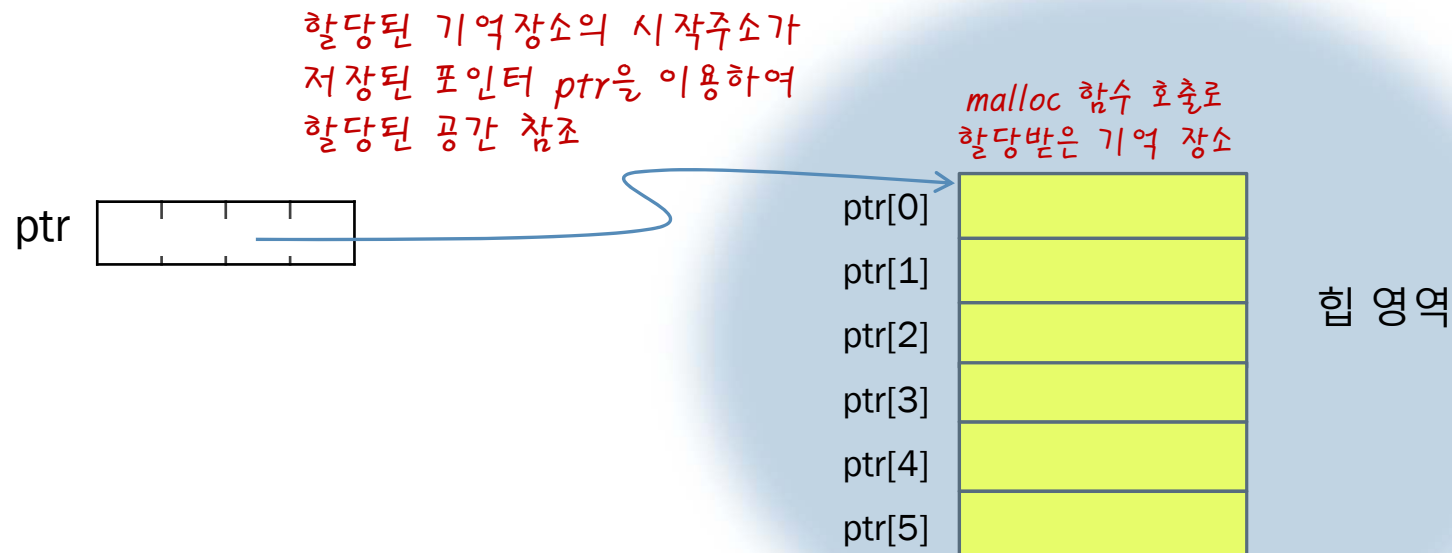


근본적인 해결

- 배열 크기를 프로그램 실행 동안 결정하고 그 만큼의 기억 공간을 할당받기
 - 동적 기억장소 할당

○ 동적 기억장소 할당이란?

- 프로그램 실행 동안 기억장소가 필요한 시점에서 필요한만큼만 요청하여 할당받는 것
- 동적 할당 요청 시 시스템에서 제공하는 기억장소는 힙 영역
- 포인터 변수와 malloc 함수를 이용 동적 할당을 이용



o malloc 함수를 이용한 동적 할당

자료형 *포인터변수명;

포인터변수명 = (자료형 *) malloc(기억장소크기);

① 동적으로 할당된 지정 크기의
기억장소의 시작 주소가 반환됨

② 시작 주소를 지정한 자료형의 포인터로 캐스팅

③ 동적할당 기억장소의 시작주소를 포인터 변수에 저장
→ 포인터변수는 할당받은 기억장소를 가리키게 됨

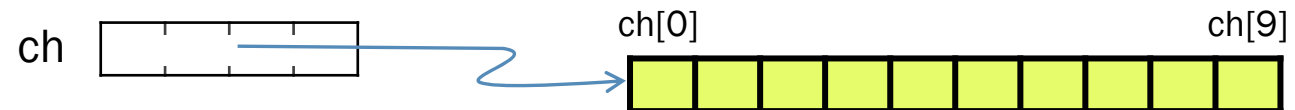
- #include <stdlib.h>가 필요
- malloc 함수는 할당받은 기억장소의 시작 주소를 void형 포인터로 반환함
→ 용도에 맞게 (포인터변수의 자료형에 맞게) 캐스팅해서 포인터변수에 저장해야 함
- 힙 영역에 할당 가능한 기억장소가 없다면 → NULL 매크로 상수가 반환됨

예)

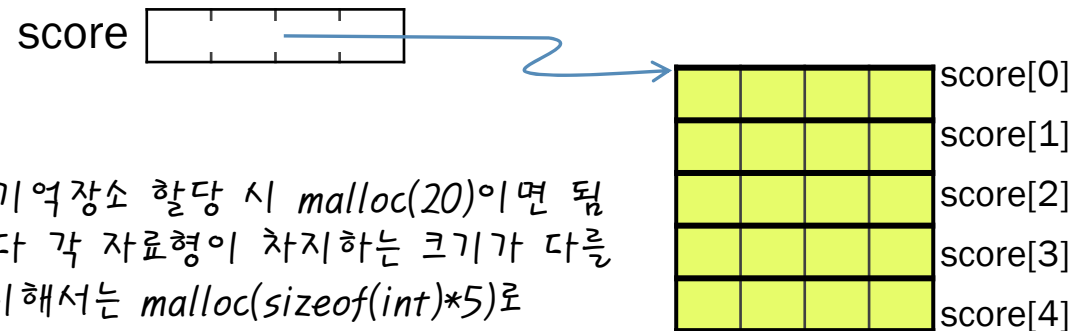
- int *ptr;
ptr = (int *) malloc(sizeof(int));



- char *ch;
ch = (char *) malloc(sizeof(char)*10);



- int *score = (int *) malloc(sizeof(int)*5);



주의

int형 정수 5개 저장용 기억장소 할당 시 malloc(20)이면 됨
그러나 컴퓨터 시스템마다 각 자료형이 차지하는 크기가 다른
수도 있으므로 이식성을 위해서는 malloc(sizeof(int)*5)로
호출하는 것이 좋다.

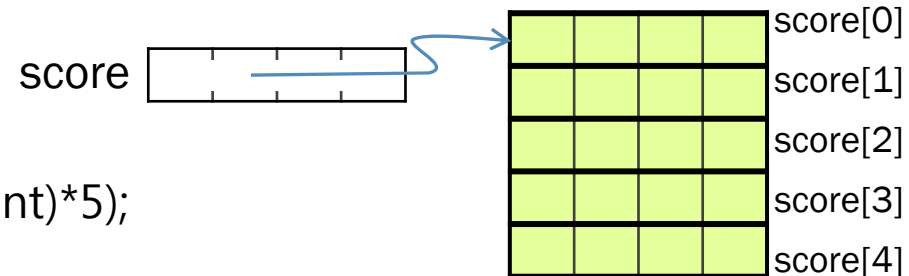
동적 할당 기억장소 해제

- 동적 할당 기억장소는 함수 실행이 끝나도 자동으로 해제되지 않음
→ 함수에서 더 이상 사용하지 않는다면 자유 상태가 되도록 해제해야 함
→ 다른 함수에서 할당받을 수 있다.

free(포인터변수명);

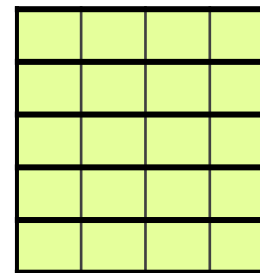
포인터변수가 가리키는 기억장소를 자유 상태로 만듦

- 예)
score = (int *) malloc(sizeof(int)*5);
:
free(score);



score

--	--	--	--



이렇게 해제된 기억장소는
시스템에 반환되며 score와의
연결이 없으므로 프로그램에서
더 이상 참조할 방법이 없다.

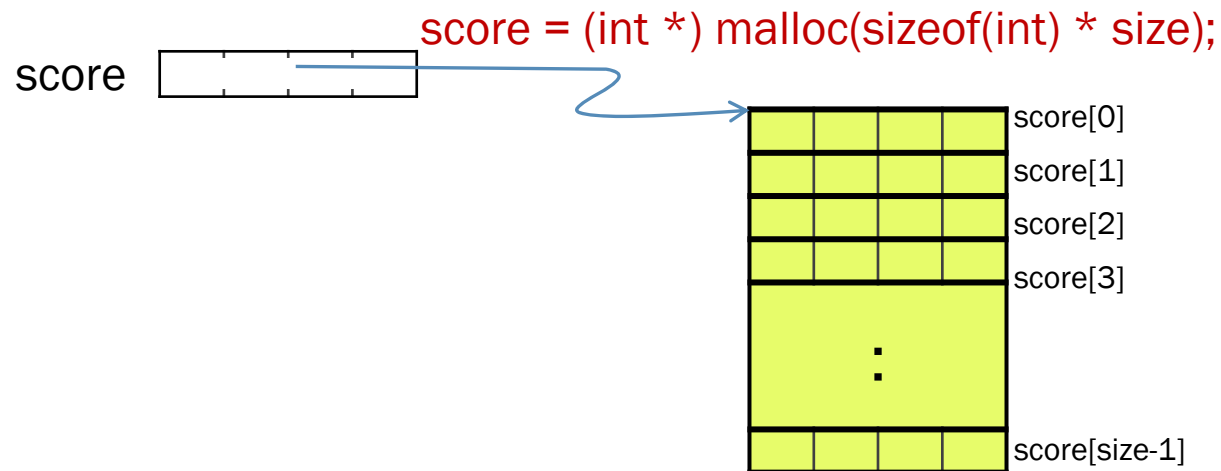
11.8.1 int형 동적 배열을 사용한 평균 처리 프로그램 p.573

문제

- 입력: 사용자가 원하는 만큼의 학생 점수
- 출력: 평균 점수

분석

- 정확한 학생수 size는 프로그램 실행 중에 사용자로부터 직접 입력받음
- size개 점수만 저장할 배열 score를 동적으로 할당받기



```

1 #include <stdio.h>
2 #include <stdlib.h> // malloc, free 함수를 위한 헤더 파일
3
4 int main()
5 {
6     int size, sum, i; // 학생수 즉 배열의 원소수
7     int *score; // 동적으로 할당받은 기억장소를 가리킬 포인터 변수 즉 배열명
8
9     // 동적 할당 기억장소의 크기를 결정하기 위해 학생 수를 입력받기
10    printf("학생수는? "); scanf("%d", &size);
11
12    // size 개수의 int형 값을 저장할 동적 기억장소 할당 후 score가 가리키기
13    score = (int *) malloc(sizeof(int) * size);
14
15    // 동적 할당을 받지 못했다면 프로그램을 강제로 끝낸다.
16    if (score == NULL)
17    {
18        printf("동적 기억장소 할당에 실패하였습니다. \n");
19        exit(1); // 프로그램 실행 중단
20    }

```

```

22 // 동적 배열에 자료 입력 후 평균을 구하여 출력하기
23 sum = 0;
24 for (i=0; i<size; i++)
25 {
26     printf("%d번째 학생의 점수는? ", i+1);
27     scanf("%d", &score[i]); // scanf("%d", score+i); 와 동일
28     sum += score[i];
29 }
30 printf("\n%d명의 평균 : %.1lf\n", size, (double)sum/size);
31
32 // score가 가리키는 동적으로 할당받은 기억장소 해제
33 free(score);
34
35 return 0;
36 }
    
```

실행결과

```

학생수는? 5
1번째 학생의 점수는? 79
2번째 학생의 점수는? 89
3번째 학생의 점수는? 98
4번째 학생의 점수는? 96
5번째 학생의 점수는? 84
    
```

5명의 평균 : 89.2

11.8.2 구조체형 동적 배열을 이용한 도서 목록 입출력 프로그램

문제

- 입력: 도서의 이름과 출판년도 정보
- 출력: 입력된 도서 정보

분석

- 도서 정보

```
struct book_info          // 구조체 템플릿 정의
{
    int year;              // 출판연도
    char title[50];        // 도서명
};
typedef struct book_info BOOK; // 자료형 재정의
```

- 입력된 도서 수 size 개의 BOOK형 구조체 저장 기억장소의 동적 할당

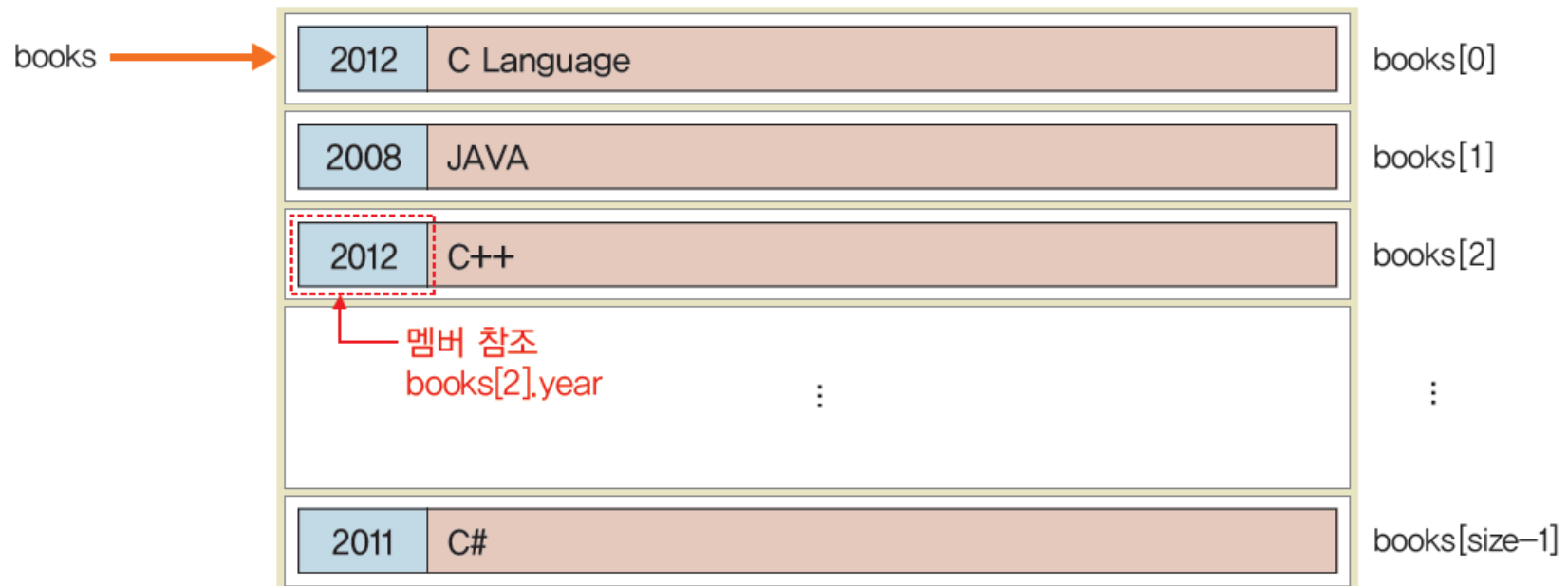
```
BOOK * books;                // 배열을 가리킬 포인터 선언
books = (BOOK *) malloc(sizeof(BOOK) * size); // 동적 할당 받기
```

11.8.2 구조체형 동적 배열을 이용한 도서 목록 입출력 프로그램

- 도서명 입력

`gets(book[i].title);` → 도서명에 포함된 빈칸도 입력하기 위해

그림 11-20 동적으로 할당받은 구조체 배열에 도서 정보가 입력된 예



```

1  #include <stdio.h>
2  #include <stdlib.h> // malloc, free, exit 함수를 위한 헤더 파일
3
4  struct book_info
5  {
6      int year;          // 출판년도
7      char title[12];    // 도서명
8  };
9  typedef struct book_info BOOK;
10
11 int main()
12 {
13     BOOK *books; // 도서 정보를 저장할 동적 배열을 가리킬 포인터
14     int size, i; // 정보를 입력받을 도서 권수
15
16     do // 도서 권수 입력(1 이상의 값 입력받기)
17     {
18         printf("정보를 입력할 도서 권수 : "); scanf("%d", &size);
19         if (size <= 0)
20             printf("에러 : 권수를 잘못 입력하였습니다. 다시 입력하세요. ");
21     } while (size <= 0);
22
23     // 도서 권수만큼 정보를 저장할 기억장소 할당받기
24     books = (BOOK *) malloc(sizeof(BOOK) * size); // 구조체 배열의 동적 할당

```

```

25  if (books == NULL) {
27      puts("동적 기억장소 할당에 실패하였습니다. \n");
28      exit(1); // 실행 중단
29  }
30
32  for (i=0; i<size; i++) // size권의 도서 정보 입력
33  {
34      printf("%d) 도서 이름 : ", i+1);
35      fflush(stdin); // 입력용 버퍼 비우기
36      gets(books[i].title);
37
38      printf(" 출판년도 : ");
39      scanf("%d", &(books[i].year));
40  }

```

실행결과

정보를 입력할 도서 권수 : 5
 1) 도서 이름 : C Language
 출판년도 : 2012
 2) 도서 이름 : JAVA
 출판년도 : 2008
 3) 도서 이름 : C++
 출판년도 : 2012
 4) 도서 이름 : PASCAL
 출판년도 : 2009
 5) 도서 이름 : C#
 출판년도 : 2011


```

42 // size권의 도서 정보 출력
43 printf("\n >> 도서 정보 목록 << \n");
44 printf("\n 순번 도서명 출판년도 ");
45 printf("\n =====\n");
46 for (i=0; i<size; i++)
47 {
48     printf("%3d %-12s %4d\n", i+1, books[i].title, books[i].year);
49 }
50 printf(" =====\n");
51
52 free(books); // 구조체 배열을 위한 동적 할당 기억장소 해제
53
54 return 0;
55 }
    
```

》 도서 정보 목록 《

순번	도서명	출판년도
1	C Language	2012
2	JAVA	2008
3	C++	2012
4	PASCAL	2009
5	C#	2011

o char형 포인터 변수를 이용한 문자열 입력: 주의점

- 반드시 문자열을 저장할 기억장소를 동적으로 할당받은 후에야 가능
- [프로그램 11-1]의 name을 char형 포인터로 선언 후 이름을 입력받도록 한다면? 다음 처럼 수정해야 함(6행을 수정하고 14행을 추가하지 않으면 에러)

```
4 struct game
5 {
6     char name[7];
7     int R1, R2, R3;
8 };
```

```
9
10 int main()
11 {
12     struct game player;
13     double avg;
14             
16     printf("선수의 이름은? ");
17     scanf("%s", player.name);
```

*char *name;*

player.name

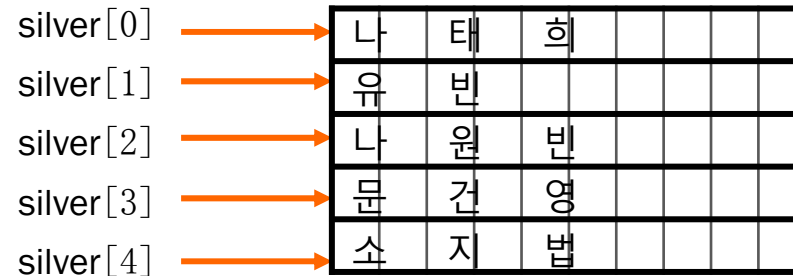


*player.name = (char *) malloc(sizeof(char) * 7);*

이름 입력 전에 길이를
어떻게 미리 알 수 있을까?

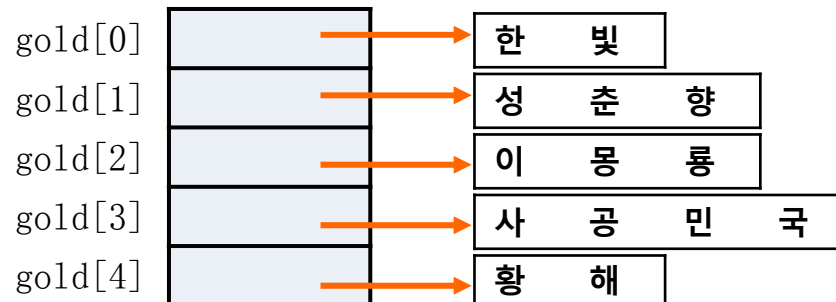
- [프로그램 10-15] 2차원 char형 배열을 이용한 문자열: 배열 공간 낭비 발생

- 7 char silver[5][10] = {"나태희", "유빈", "나원빈", "문건영", "소지법"};



- [프로그램 10-16] char형 포인터 배열을 이용한 문자열: 배열 공간 낭비 제거

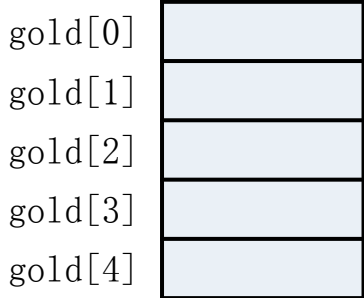
- 6 char *gold[5] = {"한빛", "성춘향", "이몽룡", "사공민국", "황해"};



문제점

- 초기화가 아니라 이름을 입력받는 경우라면?
문자열 저장용 기억장소의 동적 할당이 필요한데 그 크기는?

```
char *gold[5];  
:  
:  
for (i=0; i<5; i++)  
{  
    char *gold[i] = (char *) malloc( sizeof(char) * (이름길이 + 1) );  
    printf( " 이름: " );  
    gets(gold[i]);  
}
```



- 이름 입력 전에 길이를 어떻게 미리 알 수 있을까?
- 무조건 최대 길이로 할당받는다면 여전히 기억장소 낭비 발생
→ 동적할당이 무의미

○ 해결) 충분한 길이의 char형 배열을 임시 저장소로 사용

- 일단 사용자 이름을 임시 저장소에 저장 후 (①),
- strlen 함수를 이용 실제 저장 문자 수를 알아낸 후 이 정보를 이용하여 동적할당 받기 (②).
- 임시 저장소에 저장된 도서명을 동적으로 할당받은 기억장소에 복사 (③).

```
char temp[101]; // 최대 100개의 문자를 저장할 수 있는 임시 저장소
```

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

```
{
```

```
    printf( " 이름: ");
```

```
    gets(temp); // ①
```

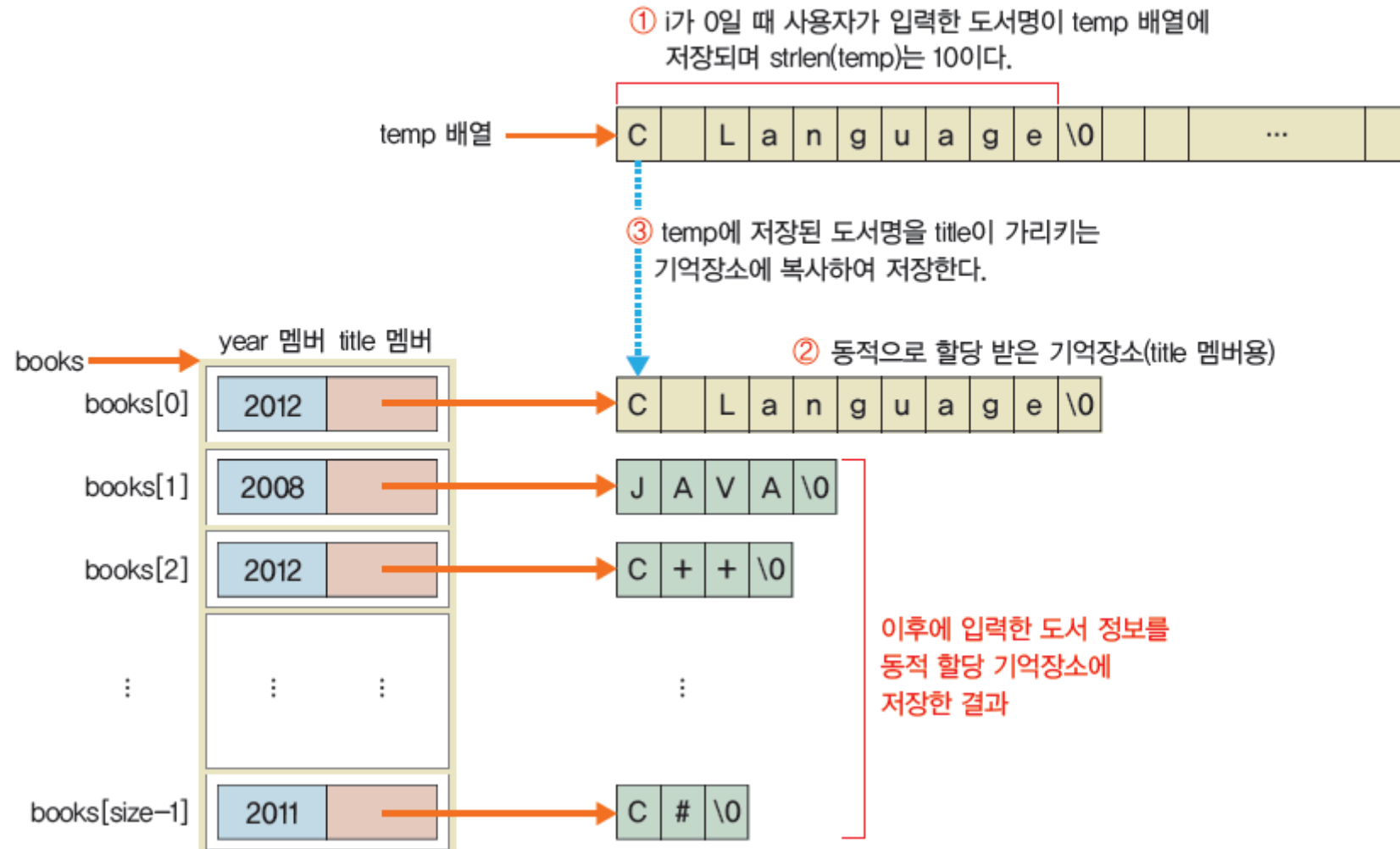
```
    gold[i] = (char *) malloc( sizeof(char) * strlen(temp) + 1 ); // ②
```

```
    strcpy(gold[i], temp); // ③
```

```
}
```

- 임시 저장소 temp의 크기가 크더라도 입력 이름이 많을수록 임시 저장소의 낭비는 무의미해짐 → 전체적으로는 효율적

그림 11-21 char형 포인터를 이용한 문자열과 동적 할당

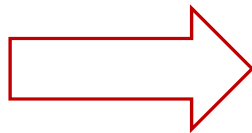


공용체

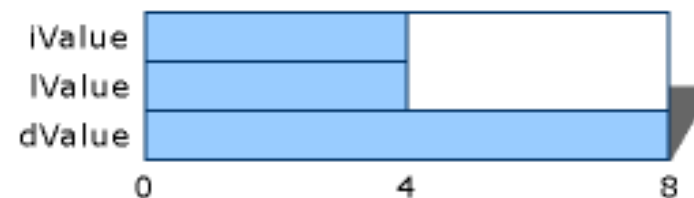
공용체(Union)

- 사용자 정의 자료형(user-defined data type)
- 다양한 자료형의 여러 값을 하나의 단위로 묶어서 관리
- 구조체(struct)와 달리 메모리를 '공유'함
- 선언된 공용체 변수 중 가장 큰 메모리를 점유하는 변수의 크기가 공용체의 전체 크기가 됨

```
union NumericType {  
    int iValue;  
    long lValue;  
    double dValue;  
};
```



Storage of Data in NumericType Union



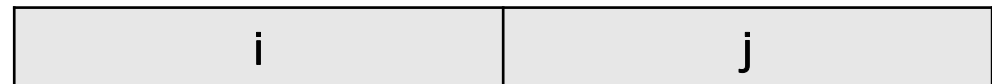
공용체

구조체의 메모리 할당

- 가장 큰 메모리를 차지하는 변수를 기준으로 그 크기가 넘어갈 경우 기준변수의 크기만큼씩 추가 할당됨

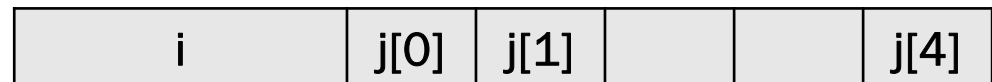
- 예 1)

```
struct A {  
    int i;  
    char j;  
}
```



- 예 2)

```
struct B {  
    int i;  
    char j[5];  
}
```



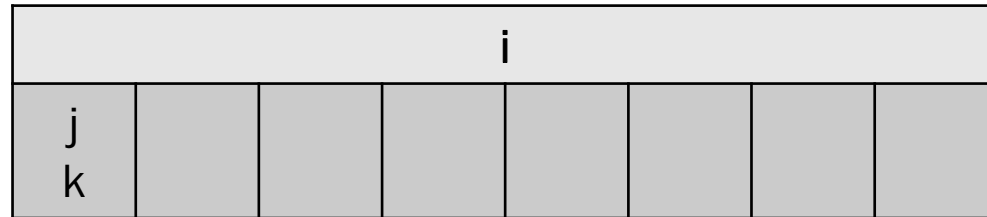
공용체

○ 공용체의 메모리 할당

- 가장 큰 메모리를 차지하는 변수의 크기 만큼 할당됨

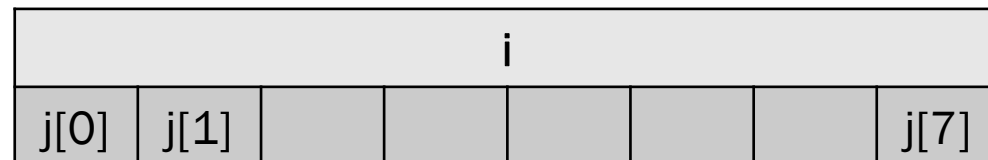
- 예 1)

```
union A {  
    int i;  
    char j;  
    char k;  
}
```



- 예 2)

```
union B {  
    int i;  
    char j[8];  
}
```



공용체

예제 (구조체 및 공용체)

```
struct A {
    int i;
    char j;
};

typedef struct A STRUCT_A;

union B {
    int i;
    char j;
    short k[2];
};

typedef union B UNION_B;

struct C {
    char s_no[10];
    char name[11];
    int grade;
    double GPA;
};
```

```
void main() {
    STRUCT_A a = { 0x12345678, 0x9 };
    UNION_B b = { 0x12345678 };
    STRUCT_C c;

    printf("STRUCT A\n");
    printf("size of STRUCT_A : %d\n", sizeof(a));
    printf("i : %x, j : %x\n\n", a.i, a.j);

    printf("UNION B\n");
    printf("size of UNION_B : %d\n", sizeof(b));
    printf("i : %x, j : %x, k1 : %x, k2 : %x\n\n",
           b.i, b.j, b.k[0], b.k[1]);

    printf("STRUCT C\n");
    printf("size of STRUCT_C : %d\n", sizeof(c));
}
```

결과는?

공용체

o Big Endian

- 낮은 주소에 상위 bit를 저장하는 방법
- 낮은 주소부터 높은 주소의 순서로 access됨

o Little Endian

- 높은 주소에 상위 bit를 저장하는 방법
- 높은 주소부터 낮은 주소의 순서로 access됨
- 우리가 일반적으로 사용하는 PC는 Little Endian을 따름

```
union B {  
    int i;  
    char j;  
    short k;  
};
```

Big Endian

100	104	108	112
12	34	56	78

Little Endian

100	104	108	112
78	56	34	12

열거형

열거형 (Enumeration)

- 순서형, 명목형 자료를 편하게 표현하기 위한 자료형
- 월, 요일, 성별 등을 숫자로 표현하면 혼동할 수 있음
이를 방지하기 위해 열거형을 사용할 수 있음
- 열거 요소의 이름만 설정할 경우, 첫 요소부터 마지막 요소까지 0부터 1씩 증가하는 값을 가지게 된다.
- 열거 요소의 값을 임의로 지정할 수 있으며, 지정되지 않은 요소는 가장 (최근에 지정된 값 + 1)을 가지게 된다.

```
enum TYPE { CHAR, INT, FLOAT, DOUBLE };  
enum DAYS { SUN, MON, TUE, WED, THU, FRI, SAT };  
enum { FALSE, TRUE };
```

```
enum TYPE { CHAR = 0, INT, FLOAT, DOUBLE };  
enum TYPE { CHAR = 3, INT, FLOAT, DOUBLE };  
enum TYPE { CHAR = 3, INT = 5, FLOAT, DOUBLE };  
enum TYPE { CHAR = 3, INT, FLOAT = 4, DOUBLE };  
enum TYPE { CHAR = 3, INT = 3, FLOAT, DOUBLE };  
enum TYPE { CHAR = 3, INT = 3, FLOAT = 4, DOUBLE = 4 };
```

```
enum TYPE t = INT;  
if(t == FLOAT) {  
    ...  
}
```

```
int a = SUN;  
switch(a) {  
    case SUN : ...;  
    case MON : ...;  
    ...  
}
```

□ typedef를 사용한 형 재정의

- typedef를 사용하여 응용자료형을 간단하게 사용할 수 있다.

```
struct student {  
    int num;  
    double grade;  
};
```

} 구조체의 형 선언

```
typedef sturct student Student; // 자료형의 재정의
```

구조체의 자료형 새로운 자료형의 이름

Student s1; // 재정의된 자료형으로 간단하게 구조체변수 선언

- 형 선언과 동시에 재정의하는 방법도 가능하다.

```
typedef struct {  
    int num;  
    double grade;  
} Student;
```

// 재정의될 것이므로 자료형의 이름이 필요 없다.
// 새로운 자료형의 이름을 바로 적어준다.

▶ typedef을 사용한 프로그램 예

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

```
typedef struct {                // 구조체의 선언과 동시에 자료형의 재정의한다.  
    int num;  
    double grade;  
} Student;
```

```
void data_prn(Student *);      // 함수의 선언, 매개변수는 Student형의 포인터변수
```

```
int main()  
{  
    Student s1={315, 4.2};     // Student형의 변수 선언과 초기화  
    data_prn(&s1);             // Student 변수의 포인터를 전달한다.  
    return 0;  
}
```

```
void data_prn(Student *sp)     // Student형을 가리키는 포인터변수  
{  
    printf("학번 : %d\n", sp->num);        // 구조체포인터변수로 멤버 참조하기  
    printf("학점 : %.1lf\n", sp->grade);  
}
```