CHAPTER

12 구조체

학습목차

학습목표

- 배열은 형태가 같은 변수가 많이 필요한 경우 유용하다.
- 형태가 서로 다른 변수들을 묶어야 한다면 구조체가 필요하다. 구조체는 새로운 자료형을 사용자가 만드는 것이고, 한번 형태가 정의되면 그 이후부터는 구조체 변수, 구조체 배열, 구조체 포인터 등으로 쉽게 활용할 수 있다.

12 같이 만들어 볼 WONDER 코드

문자 비 코드 동작 화면

문자 이동 코드 동작 화면

12.1 구조체 정의

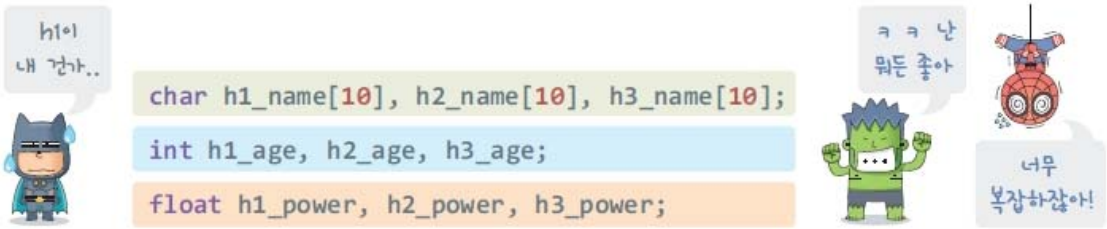
1 구조체란?

- 같은 크기와 종류의 자료형을 하나로 묶은 것은 배열
- 구조체(structure)는 서로 다른 자료형을 하나로 묶어 놓은 것

12.1 구조체 정의


```
char h1_name[10], h2_name[10], h3_name[10];
int h1_age, h2_age, h3_age;
float h1_power, h2_power, h3_power;
```

이름을 저장하는 문자열 변수
나이를 저장하는 정수형 변수
힘을 저장하는 실수형 변수




```
char h1_name[10], h2_name[10], h3_name[10];
int h1_age, h2_age, h3_age;
float h1_power, h2_power, h3_power;
```

12.1 구조체 정의

2 구조체 정의 및 선언

- 구조체는 사용자가 정의하는 자료형

- 구조체의 이름과 구조체 안에 어떤 자료형들이 들어가야 하는지를 결정하는 단계가 구조체 정의(structure definition)이며, 자신이 정의한 구조체의 특징을 가지는 변수를 선언하는 단계가 구조체 변수 선언 단계
- 구조체 정의 방법

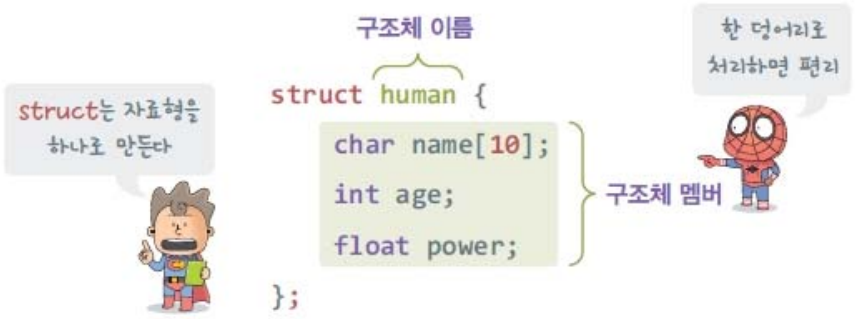
구조체 정의 방법

```
struct 구조체_이름 {
    자료형 변수명;
    자료형 변수명;
};
```

구조체 정의 시작
멤버 1
멤버 2
구조체 정의 끝은 중괄호와 세미콜론!

12.1 구조체 정의

- 구조체 정의



- 구조체 정의는 사용자 정의 자료형의 틀을 만드는 작업

12.1 구조체 정의

구조체 변수 선언 방법

```
struct 구조체_이름 변수이름;
```

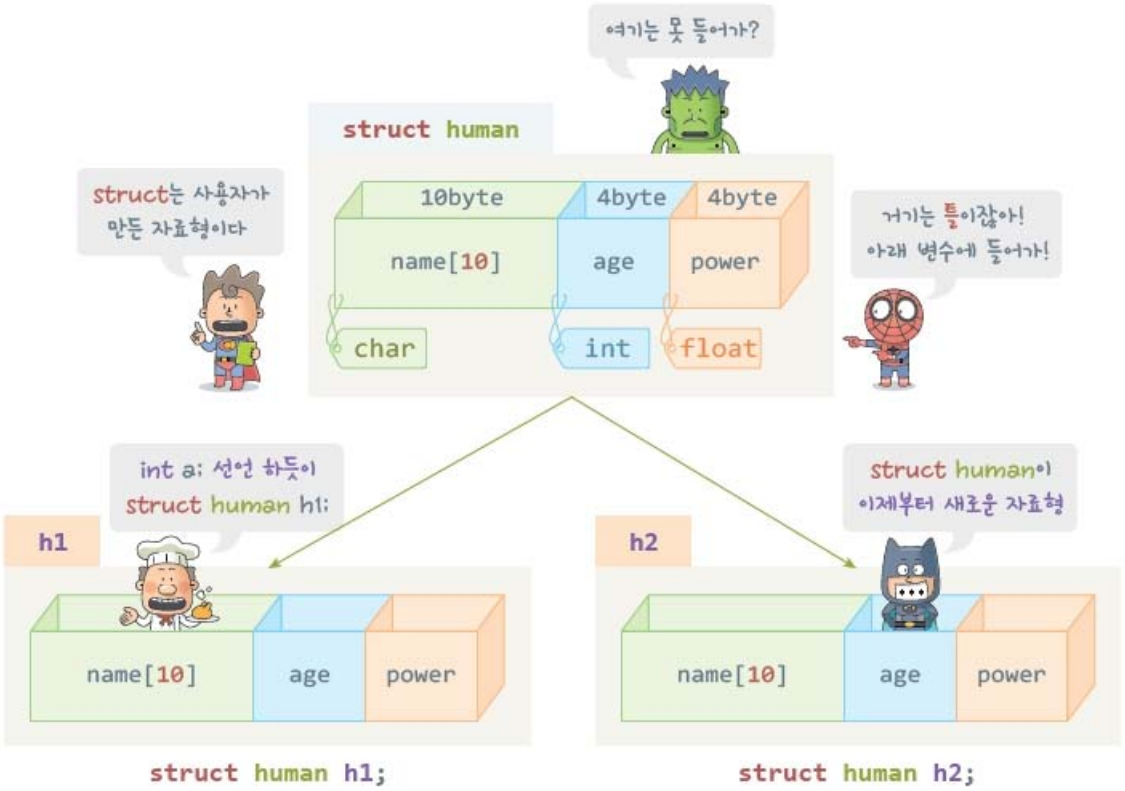
```
struct human {
    char name[10];
    int age;
    float power;
};

int main() {
    struct human h1;
```

```
human 구조체 정의
    human 구조체 멤버 name
    human 구조체 멤버 age
    human 구조체 멤버 power
human 구조체 정의 끝 - 세미콜론 필수


    human 구조체 변수 h1 선언
```


12.1 구조체 정의

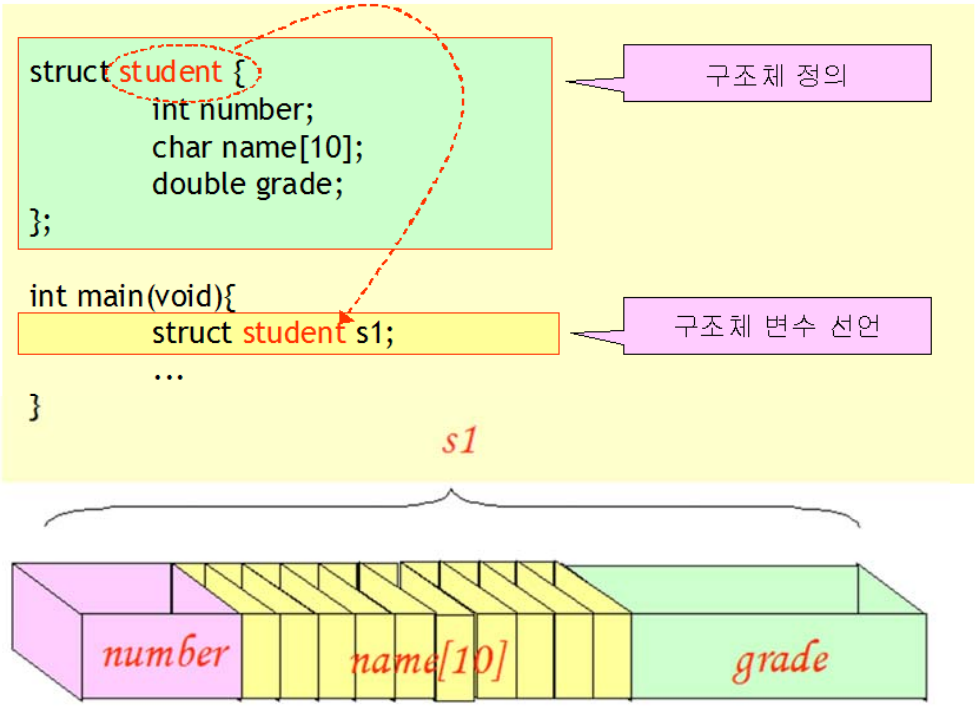
> 구조체 변수 선언

- 일반적으로 구조체 정의는 main() 함수 밖에서 하고
- 구조체 변수를 만드는 것은 main() 함수 안에서



구조체 변수 선언

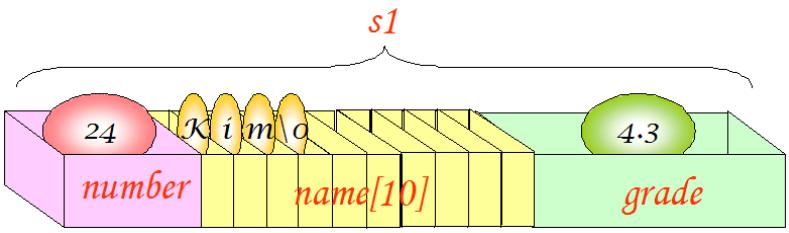
- 구조체 정의와 구조체 변수 선언은 다르다.



구조체의 초기화

- 중괄호를 이용하여 초기값을 나열한다.

```
struct student {
        int number;
        char name[10];
        double grade;
};
struct student s1 = { 24, "Kim", 4.3 };
```



3 구조체에 데이터 넣기

- 점(.) 연산자를 사용하여 "구조체변수.멤버"에게 접근
- human 구조체 변수 h1의 나이(age)에 22라는 값을 넣고 싶다면 h1.age = 22라고 쓰면 됨
- h1.name에 gildong을 넣고 싶다면 문자열 복사 함수인 strcpy()를 사용

```
struct human h1;                          구조체 변수 h1 선언

   h1.age = 22;                           구조체 변수.멤버에 값 대입
   strcpy(h1.name, "gildong");            멤버에 문자열 복사
```


[예제 12-1] 구조체 선언과 사용 코드

```
01  #include <stdio.h>
02  #include <string.h>
03
04  struct human {                          // human 구조체 정의
05     char name[10];                          // human 구조체 멤버 name
06     int age;                                // human 구조체 멤버 age
07     float power;                            // human 구조체 멤버 power
08  };                                      // human 구조체 정의 끝 - ;필수
09
10  int main() {
11     struct human h1;                        // human 구조체 변수 h1 선언
12
13     strcpy(h1.name, "gildong");             // h1.name에 문자열 gildong 복사
14     h1.age = 22;                            // h1.age에 정수 22 대입
15     h1.power = 8.2;                         // h1.power에 실수 8.2 대입
16
17     printf("name = %s\n", h1.name);
18     printf("age = %d\n", h1.age);
19     printf("power = %.1f\n", h1.power);
20     return 0;
21  }
```

실행 화면

```
name = gildong
age = 22
power = 8.2
```

- 여러 구조체 변수 선언 방법

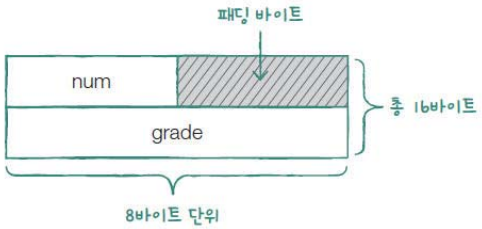
```
struct human h1, h2, h3;
```
human 구조체 변수 h1, h2, h3 선언

```
struct human {
    char name[10];
    int age;
    float power;
} h1, h2, h3;
```
human 구조체 선언

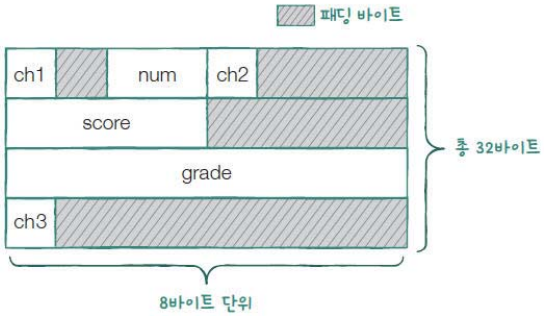
human 구조체 변수 h1, h2, h3 선언(전역 변수)

- **구조체 변수의 크기**
  - 구조체는 실행 효율을 위해 패딩 바이트를 넣어 바이트 정렬을 한다.
  - double이 없으면 기본적으로는 sizeof(int)로 정렬, 문자 배열도 sizeof(int)로 끝을 맞춤

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




```
struct student
{
    char ch1;
    short num;
    char ch2;
    int score;
    double grade;
    char ch3;
};
```



12.1 구조체 정의

[예제 12-2] 구조체 변수 선언과 초기화 코드

```c
#include <stdio.h>

struct human {                          // human 구조체 정의
   char name[10];                          // human 구조체 멤버: 이름(문자열)
   int age;                                // human 구조체 멤버: 나이(정수)
   float power;                            // human 구조체 멤버: 키(실수)
};                                      // human 구조체 정의 끝

int main() {
   struct human h1 = {"gildong", 22, 8.2}, h2 = {"hyungwook", },
                h3 = {NULL};

   printf("h1:%s, %d, %.1f\n", h1.name, h1.age, h1.power);
   printf("h2:%s, %d, %.1f\n", h2.name, h2.age, h2.power);
   printf("h3:%s, %d, %.1f\n", h3.name, h3.age, h3.power);
   return 0;
}
```

실행 화면

```
h1:gildong, 22, 8.2
h2:hyungwook, 0, 0.0
h3:, 0, 0.0
```


01 다음 중 사용자가 정의하는 자료형은 무엇인가?

① 정수 ② 배열 ③ 실수 ④ 구조체

02 다음 중 서로 다른 자료형을 묶어 하나의 자료형으로 만든다는 뜻을 가진 단어는 무엇인가?

① 정수 ② 배열 ③ 실수 ④ 구조체

03 다음 중 구조체의 이름과 구조체 안에 어떤 자료형들이 들어가야 하는지를 결정하는 단계는 무엇인가?

① 구조체 정의 ② 자료형 별명 ③ 구조체 변수선언 ④ 구조체 포인터

04 다음 중 자신이 정의한 구조체의 특징을 가지는 변수를 선언하는 단계는 무엇인가?

① 구조체 정의 ② 구조체 멤버 ③ 구조체 변수선언 ④ 구조체 포인터

05 다음 중 구조체의 안에 정의된 자료형을 가리키는 단어는 무엇인가?

① 구조체 정의 ② 구조체 멤버 ③ 구조체 변수선언 ④ 구조체 포인터

06 다음 중 구조체 안의 멤버에게 접근할 때 사용하는 기호는 무엇인가?

① 점(.) ② 느낌표(!) ③ 별(*) ④ 쉼표(,)

07 다음 중 구조체 변수를 여러 개 선언할 때 사용하는 기호는 무엇인가?

① 점(.) ② 느낌표(!) ③ 별(*) ④ 쉼표(,)

08 다음 중 구조체를 정의할 때 사용하는 단어는 무엇인가?

① `struct` ② `enum` ③ `typedef` ④ `union`

18 사용자가 정의하는 자료형이 ( )이다.

19 서로 다른 자료형을 묶어 하나의 자료형으로 만든다는 뜻을 가진 단어는 ( )이다.

20 구조체의 이름과 구조체 안에 어떤 자료형들이 들어가야 하는지를 결정하는 단계를 ( ) 단계라 부른다.

22 구조체 안에 정의된 자료형을 ( )라 부른다.

23 구조체 안의 멤버에게 접근할 때 사용하는 기호는 ( )이다.

24 구조체 변수를 여러 개 선언할 때 사용하는 기호는 ( )이다.

25 구조체를 정의할 때 사용하는 단어는 ( )이다.

마무리

❖ 크래커의 가격(price)과 열량(calories)을 저장할 cracker 구조체를 선언한다. 만든 구조체로 변수를 선언하고 가격과 열량을 키보드로 입력한 후에 화면으로 출력하는 프로그램을 작성하라.

// 실행결과
바사삭의 가격과 열량을 입력하세요 : 1200 500
바사삭의 가격 : 1200원
바사삭의 열량 : 500kcal

12.2 사용자 정의 자료형

1 사용자 정의 자료형 만들기 : typedef

- 기존 자료형의 별명을 만들어 주는 키워드가 typedef
- unsigned int를 UINT라는 별명으로 바꾸고 싶다면, typedef unsigned int UINT로 정의한 후 사용
- UINT num1이라고 변수를 선언하면 num1은 unsigned int형 변수가 됨

기존 자료형의 별명 만들기

```
typedef 기존_자료형 별명;
```

```
typedef unsigned int UINT;
typedef long long llong;

int main() {
   UINT num1;
   llong num2;
```

```
UINT는 unsigned int의 별명
llong은 long long의 별명


UINT와 unsigned int 둘 다 사용 가능
llong과 long long 둘 다 사용 가능
```

```
typedef int* INT_P;

int main() {
   INT_P num1;
```

```


별표(*)가 없어 일반 변수처럼 보임
```

12.2 사용자 정의 자료형

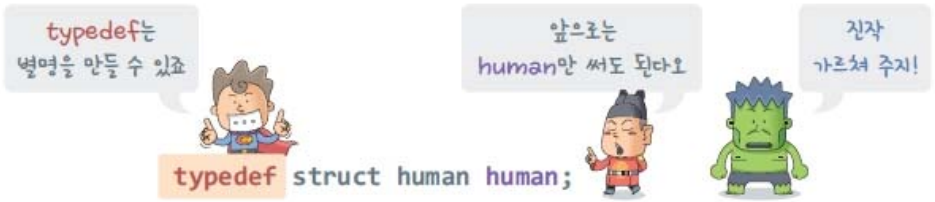
- 구조체를 단순하게 만들 때 typedef 사용

```
typedef struct human human;

int main() {
    struct human h1;
    human h2;
```

human는 struct human의 별명

기존 방법 사용 가능

struct를 생략해도 human 변수 선언 가능



12.2 사용자 정의 자료형

```
typedef struct Human {
    char name[10];
    int age;
    float power;
} human;

int main() {
    struct Human h1;
    human h2;
```

typedef를 사용하여 Human 구조체 정의

구조체 정의 시 typedef를 사용할 때 앞쪽의 Human 생략 가능

새로운 구조체 자료형 이름은 human

기존 방법도 가능

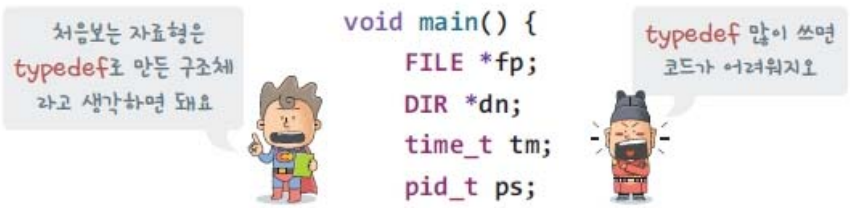
struct를 생략해도 human 구조체 변수 선언 가능

```
typedef struct human {
    char name[10];
    int age;
    float power;
} human;
```

typedef를 사용하여 human 구조체 정의

구조체 정의 시 typedef를 사용할 때 앞쪽의 human 생략 가능

별명이 구조체 이름과 같아도 됨



2 구조체 복사하기

[예제 12-3] 구조체 복사 코드

```c
#include <stdio.h>

typedef struct human {              // human 구조체 정의
   char name[10];
   int age;
   float power;
} human;                            // human 구조체 별명

int main() {
   human h1, h2;                       // typedef로 struct 없이 변수 만듦

   printf("이름 나이 힘을 순서대로 입력: ");
   scanf("%s %d %f", h1.name, &h1.age, &h1.power);

   h2 = h1;                            // 구조체 h1 값을 h2로 복사
   printf("h1:%s, %d, %.1f\n", h1.name, h1.age, h1.power);
   printf("h2:%s, %d, %.1f\n", h2.name, h2.age, h2.power);
   return 0;
}
```

실행 화면

```
이름 나이 힘을 순서대로 입력: hyungwook 28 9.3
h1:hyungwook, 28, 9.3
h2:hyungwook, 28, 9.3
```


- 구조체 연산의 주의사항

```c
if ( h1 == h2 )
   printf("둘이 같다");
```

오류 - 구조체의 직접 비교는 허용 안 함

```c
if ( h1.age == h2.age )
   printf("둘이 친구");
```

구조체 멤버들끼리는 비교 가능

12.2 사용자 정의 자료형

3 멤버 구조체

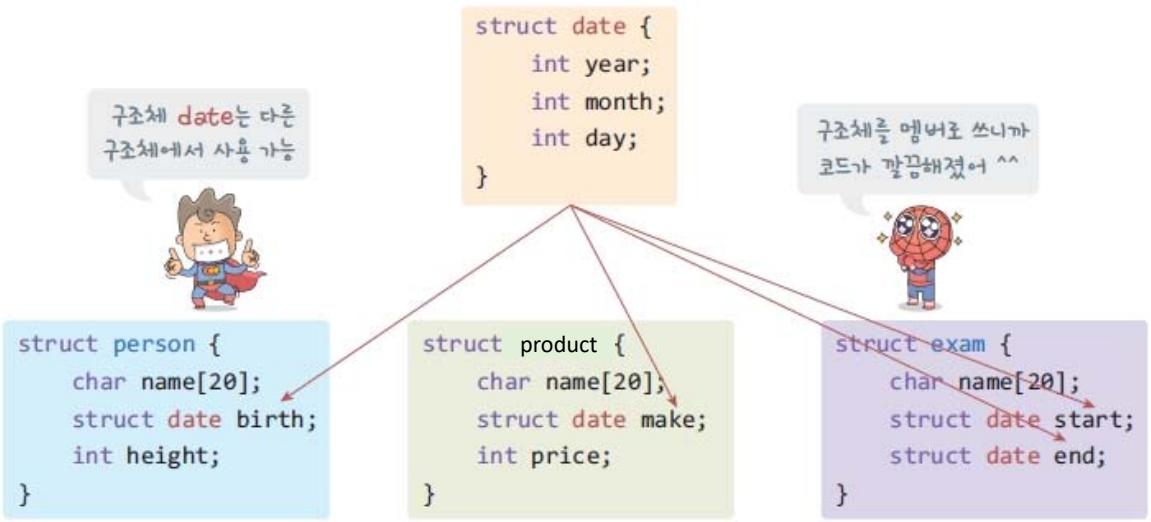
- 멤버 구조체 – 구조체 안에 다른 구조체 사용

```
struct date{
    int year;
    int month;
    int day;
};

struct human {
    char name[10];
    struct date birth;
};

int main() {
    struct human h1;
```

```
date 구조체 정의 – 사용 전에 먼저 정의되어야 함
    date 구조체 멤버 : year(정수)
    date 구조체 멤버 : month(정수)
    date 구조체 멤버 : day(정수)


human 구조체 정의

    생일을 나타내는 birth는 date 구조체 사용
```


12.2 사용자 정의 자료형

- 멤버 구조체 – 구조체의 재사용



- 멤버 구조체 접근 방법

```
strcpy(h1.name,"gildong");
h1.birth.year = 1989;
h1.birth.month = 2;
h1.birth.day = 11;
```

```
birth에 속한 year에 접근 : birth.year
birth에 속한 month에 접근 : birth.month
birth에 속한 day에 접근 : birth.day
```

[예제 12-4] 족보 정리 코드

```c
#include <stdio.h>

struct date{                            // date 구조체 정의
   int year, mon, day;
};

struct human {                          // human 구조체 정의
   char name[10];
   struct date bt;                      // bt(birth)는 date 구조체 사용
};

int main() {
   struct human h1, h2;
   int jud = 0;                            // 누가 형인지 판단

   printf("사람1-이름, 태어난 년도, 월 입력 : ");
   scanf("%s %d %d", h1.name, &h1.bt.year, &h1.bt.mon);
   printf("사람2-이름, 태어난 년도, 월 입력 : ");
   scanf("%s %d %d", h2.name, &h2.bt.year, &h2.bt.mon);

   if (h1.bt.year < h2.bt.year) jud = 1;
   else if (h1.bt.year > h2.bt.year) jud = 2;
   else if (h1.bt.year == h2.bt.year && h1.bt.mon < h2.bt.mon) jud = 1;
   else if (h1.bt.year == h2.bt.year && h1.bt.mon > h2.bt.mon) jud = 2;

   if (jud == 1) printf("%s 형님\n", h1.name);      // h1이 형님
   else if (jud == 2) printf("%s 형님\n", h2.name); // h2가 형님
   else printf("둘은 친구\n");                      // jud = 0 둘은 친구
   return 0;
}
```

실행 화면

```
사람1-이름, 태어난 년도, 월 입력 : changsik 2000 4
사람2-이름, 태어난 년도, 월 입력 : gildong 2000 2
gildong 형님
```


마무리

❖ 다음과 같이 구조체가 선언되어 있을 때, 잘못 사용한 것을 찾아 이유를 설명하라.

```c
struct profile {
      int num;
      char name[20];
      char *skill;
};
struct sports {
      char *event;
      struct profile player;
};
struct sports a;
```

① strcpy(a.event, “figure skating”);
② a.player.name = “Yuni Seo”;
③ a.num = 19;
④ scanf(“%s”, a.player.skill);

12.3 구조체 배열과 포인터

1 구조체 배열

- 구조체 배열은 일반적인 배열과 똑같음
- 구조체 변수 뒤에 대괄호[]를 사용하여 배열을 만들면 됨

```
struct human {
   char name[10];
   int age;
   float power;
};

int main() {
   struct human ho[3];
```

사람 구조체 배열 ho[3] 선언

12.3 구조체 배열과 포인터

[예제 12-5] 구조체 배열 코드

```
01  #include <stdio.h>
02
03  struct human {                          // human 구조체 정의
04     char name[10];                          // human 구조체 멤버: 이름(문자열)
05     int age;                                // human 구조체 멤버: 나이(정수)
06     float power;                            // human 구조체 멤버: 힘(실수)
07  };                                      // human 구조체 정의 끝
08
09  int main() {
10     struct human ho[3] = { {"gildong", 22, 8.2},
11                            {"hyungwook", 28, 9.3},
12                            {"changsik", 21, 7.6} };
13     int k;
14
15     for (k = 0; k < 3; k++)
16        printf("%d:%s %d %.1f\n", k, ho[k].name, ho[k].age, ho[k].power);
17     return 0;
18  }
```

실행 화면

```
0:gildong 22 8.2
1:hyungwook 28 9.3
2:changsik 21 7.6
```

2 구조체 배열 사용하기

[예제 12-6] 구조체 배열값 바꾸기 코드

```c
#include <stdio.h>

typedef struct human {                  // human 구조체 정의
   char name[10];                          // human 구조체 멤버: 이름(문자열)
   int age;                                // human 구조체 멤버: 나이(정수)
   float power;                            // human 구조체 멤버: 키(실수)
} human;                                // human 구조체 정의 끝

int main() {
   human ho[3] = { {"gildong", 22, 8.2}, {"hyungwook", 28, 9.3},
                   {"changsik", 21, 7.6} };
   human temp;
   int k;

   temp = ho[0];
   ho[0] = ho[2];
   ho[2] = temp;

   for (k = 0; k < 3; k++)
      printf("%d:%s %d %.1f\n", k, ho[k].name, ho[k].age, ho[k].power);
   return 0;
}
```

실행 화면

```
0:changsik 21 7.6
1:hyungwook 28 9.3
2:gildong 22 8.2
```


[예제 12-7] 출동순서 정하기 코드: 오류 수정하기

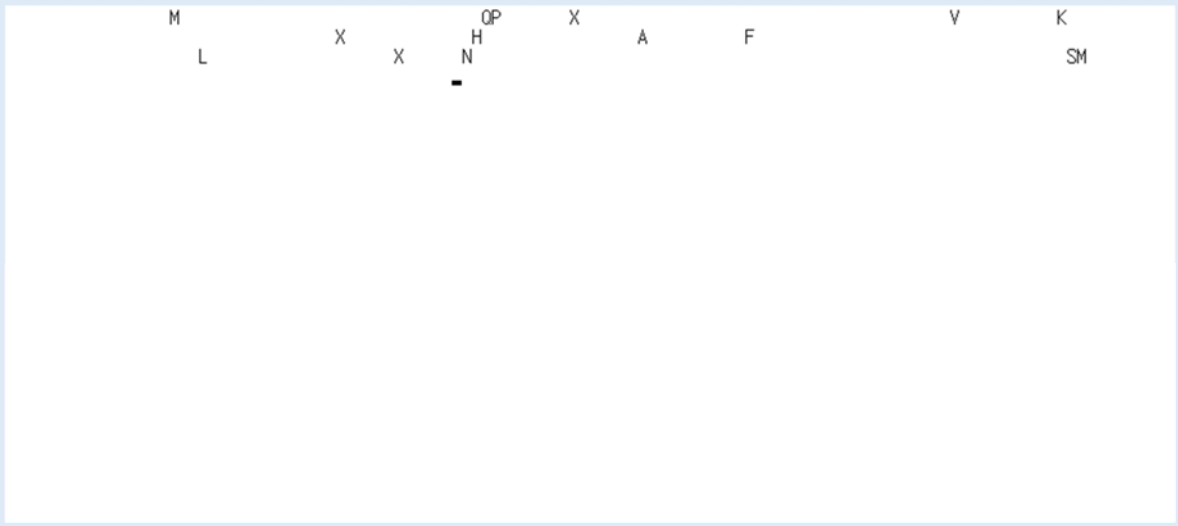
```c
#include <stdio.h>

#define MAX 4

typedef struct human {            // typedef를 사용하여 human 구조체 정의
   char name[10];                    // human 구조체 멤버: 이름(문자열)
   int age;                          // human 구조체 멤버: 나이(정수)
   float power;                      // human 구조체 멤버: 힘(실수)
} human;                          // human 구조체 별명 human

int main() {
   human ho[MAX] = { {"gildong", 22, 8.2}, {"hyungwook", 28, 9.3},
                     {"changsik", 21, 7.6}, {"hosik", 29, 8.1} }, temp;
   int j, k;

   for (k = 0; k < MAX; k++) {        // for k - 구조체 배열 수만큼 반복
      for (j = 1; j < MAX; j++) {        // for j - 두 수 비교
         if (ho[j-1].age > ho[j].age) {     // if 나이가 많으면
            temp = ho[j-1];                    // 두 구조체 바꾸기
            ho[j-1] = ho[j];
            ho[j] = temp;
         }
      }                                  // End for j
   }                                  // End for k
   for (k = 0; k < MAX; k++)
      printf("%d:%s %d %.1f\n", k, ho[k].name, ho[k].age,
                                                  ho[k].power);
   return 0;
}
```

실행 화면

```
0:changsik 21 7.6
1:gildong 22 8.2
2:hyungwook 28 9.3
3:hosik 29 8.1
```

LAB 1 문자 비 코드

문자 비 코드 동작 화면



12.3 구조체 배열과 포인터

[예제 12-8] 문자비 코드

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

typedef struct mych {                  // mych 구조체 정의
   char ch;                               // 출력 문자를 저장하는 멤버
   int pos;                               // 문자를 위치를 저장하는 멤버
} mych;                                // mych 구조체 별명(구조체 이름과 같음)

int main() {
   int k, jul, met;
   mych rain[118];                        // 구조체 mych 배열 선언

   for (met = 0; met < 118; met++) {
         rain[met].pos = rand() % 26 - 26;
         rain[met].ch = rand() % 25 + 65;
   }
   for (k = 0; k < 200; k++) {
      system("cls");
      for (met = 0; met < 118; met++)
         if (rain[met].pos < 26) rain[met].pos++;
         else rain[met].pos = -1;
      for (jul = 0; jul < 26; jul++) {
         for (met = 0; met < 118; met++) {
            if (rain[met].pos == jul) printf("%c", rain[met].ch);
            else printf(" ");
         }
         printf("\n");
      }                                       // End of for jul
   }                                        // End of for k
   return 0;
}
```

12.4 구조체와 포인터

1 구조체 포인터 이해하기

- 핸드폰에서 사용할 전화번호부를 만드는 경우 배열이 가장 쉬운 방법
  - 배열로 만들 경우 배열의 크기를 잡기 어려움
  - 가나다 순으로 정렬하기 위해서 추가 작업을 해주어야 함
  - 사용자가 전화번호부 내용을 지웠을 경우, 다른 내용으로 채워 질 때 까지 빈자리로 남게 됨

12.4 구조체와 포인터

- 구조체 포인터는 구조체의 주소를 저장하는 자료형이다. 일반적인 포인터와 같이 포인터 연산자(*)를 붙여서 선언
- 연결 리스트(linked list)를 많이 사용하는데, 연결 리스트의 핵심 자료구조가 구조체 포인터
- 구조체 포인터를 연결 리스트에 사용하기 위해서는 동적 메모리 할당을 배워야 함 - 동적 메모리 할당은 뒤에서 자세히 다룰 예정
- 이 절에서는 구조체 포인터를 선언하고 멤버에 접근하는 방법만을 설명

```
struct human {
    char name[10];
    int age;
    float power;
};

int main() {
    struct human h1 = {"gildong", 22, 8.2};
    struct human *sp;
    sp = &h1;
```

구조체를 가리키는 포인터 *hp 선언
구조체 h1 주소를 포인터 hp에게 줌

12.4 구조체와 포인터

- 포인터형 구조체의 사용방법은 포인터 사용방법과 같음
- 주소 변환 연산자(&)를 사용하여 sp = &h1와 같이 h1이 자신의 주소를 포인터 sp에게 주면, sp는 h1에 접근 할 수 있게 됨
- 자신의 주소를 넘겨준다는 것은 포인터 sp에세 h1의 데이터를 맘대로 할 수 있는 권리를 넘겨 준 것

- 기존의 점(.)을 사용하는 방법으로 (*sp).name이라 쓰면 멤버 name에 접근 할 수 있음
- 포인터를 이용하여 구조체 멤버에 접근할 때에는 점(.)이 아닌 화살표(->) 연산자를 사용하는 것이 편리

포인터형 구조체의 멤버 참조 방법

(*sp).name	멤버 name 참조
sp->name	멤버 name 참조

12.4 구조체와 포인터

[예제 12-9] 구조체 포인터 코드

```c
#include <stdio.h>

struct human {                          // human 구조체 정의
    char name[10];                          // human 구조체 멤버: 이름(문자열)
    int age;                                // human 구조체 멤버: 나이(정수)
    float power;                            // human 구조체 멤버: 힘(실수)
};                                      // human 구조체 정의 끝

int main() {
    struct human h1 = {"gildong", 22, 8.2};
    struct human *sp;                       // 구조체를 가리키는 포인터 *sp 선언

    sp = &h1;                               // 구조체 h1 주소를 포인터 sp에게 줌
    printf("%s %d %.1f\n", (*sp).name, (*sp).age, (*sp).power);
    printf("%s %d %.1f\n", sp->name, sp->age, sp->power);
    return 0;
}
```

실행 화면

```
gildong 22 8.2
gildong 22 8.2
```

2 포인터를 멤버로 가지는 구조체

- 구조체의 멤버로 포인터를 가지는 경우
- struct date *bt; 로 선언하여 사용하면 됨

```
struct date{
   int year, mon, day;
};
 struct human {
   char name[10];
   struct date *bt;
};
```

date 구조체 정의 – 사용 전에 먼저 정의되어야 함

human 구조체 정의

bt는 date 구조체를 가리키는 포인터

[예제 12-10] 포인터를 멤버로 가지는 구조체 코드

```
01  #include <stdio.h>
02
03  struct date{                          // date 구조체 정의
04     int year, mon, day;
05  };
06
07  struct human {                        // human 구조체 정의
08     char name[10];
09     struct date *bt;                   // bt는 date 구조체를 가리키는 포인터
10  };
11
12  int main() {
13     struct human h1 = {"gildong"};
14     struct date d1 = {1989, 12, 12};    // d1은 date 구조체를 만들고 초기화
15
16     h1.bt = &d1;                       // h1.bt에 d1의 주소 대입
17     printf("이름: %s\n", h1.name);
18     printf("%d년 %d월 %d일", h1.bt->year, h1.bt->mon, h1.bt->day);
19     return 0;
20  }
```

실행 화면

```
이름: gildong
1989년 12월 12일
```

12.4 구조체와 포인터

- 구조체가 멤버로 포인터를 가지는 가장 큰 이유는 연결 리스트(linked list)를 만들기 위해서임
- 자기 자신을 가리킬 수 있는 포인터를 가진 구조체는 다음과 같이 정의 함

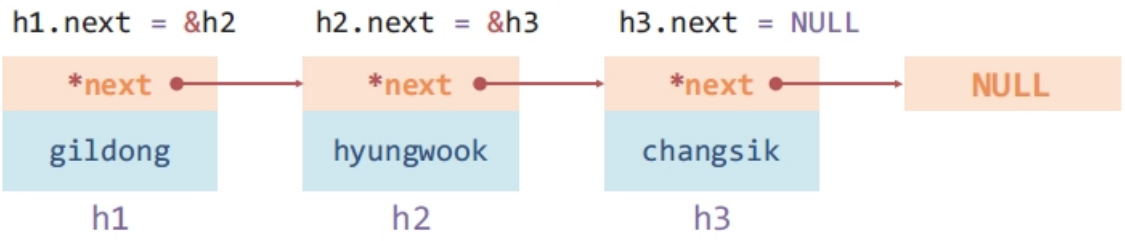
자신을 가리키는 구조체 정의 방법

```
typedef struct human {
    char name[10];
    struct human *next;
} human;
```

typedef를 사용하여 human 구조체를 정의해야 함

struct를 사용하여 human *next를 정의해야 함

- 구조체 포인터를 사용한 연결 리스트 구조



12.4 구조체와 포인터

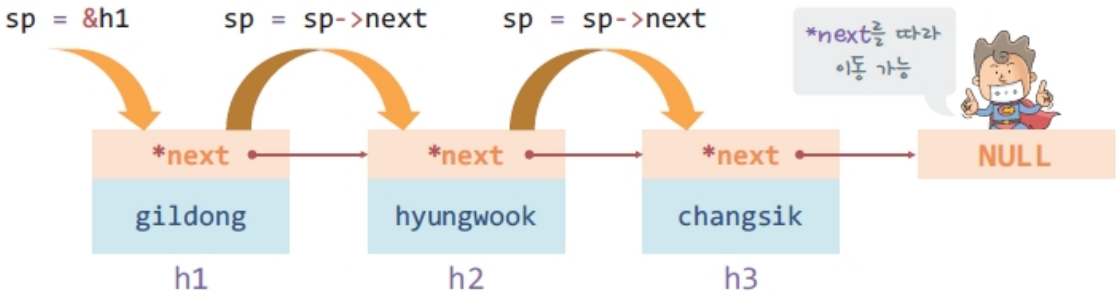
[예제 12-11] 자기 자신을 가리키는 포인터 구조체 코드: 중요

```
01  #include <stdio.h>
02
03  typedef struct human {        // typedef를 사용하여 human 구조체를 정의해야 함
04      char name[10];
05      struct human *next;       // struct를 사용하여 human *next를 정의해야 함
06  } human;
07
08  int main() {
09      human h1 = {"gildong"}, h2 = {"hyungwook"}, h3 = {"changsik"};
10      human *sp;
11
12      h1.next = &h2;
13      h2.next = &h3;
14      h3.next = NULL;
15
16      sp = &h1;
17      printf("%s\n", sp->name);
18      sp = sp->next;
19      printf("%s\n", sp->name);
20      sp = sp->next;
21      printf("%s\n", sp->name);
22      return 0;
23  }
```

실행 화면

```
gildong
hyungwook
changsik
```

- 연결 리스트에서 이동 방법



LAB 2 문자 이동 코드

문자 이동 코드 동작 화면

LAB 2 문자 이동 코드

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

typedef struct mych {                  // mych 구조체 정의
   char ch;                            // 출력문자를 저장하는 멤버
   int pos, spd;                       // 위치와 속도를 저장하는 멤버
} mych;                                // mych 구조체 별명(구조체 이름과 같음)

int main() {
   int met, k, jul;                    // pos[] 가로 위치, spd[] 이동속도
   mych cld[26];                       // 구조체 mych 배열 선언

   for (jul = 0; jul < 26; jul++) {
      cld[jul].ch = rand() % 25 + 65;
```


LAB 2 문자 이동 코드 (계속): 문자가 그려지는 조건은?

```c
      cld[jul].pos = rand() % 26 - 26;          // 화면 밖에서 시작
      cld[jul].spd = rand() % 5 + 1;            // 속도는 1 - 5 사이
   }
   for (k = 1; k < 100; k++) {
      system("cls");
      for (jul = 0; jul < 26; jul++) {
         cld[jul].pos = cld[jul].pos + cld[jul].spd;
         if (cld[jul].pos > 110) cld[jul].pos = rand() % 26 - 26;
         for (met = 0; met <= 110; met++) {
            if (cld[jul].pos == met) printf("%c", cld[jul].ch);
            else printf(" ");
         }
           printf("\n");
      }                                         // End of for jul
   }                                          // End of for k
   return 0;
}
```

12.5 구조체와 함수

1 구조체와 매개 변수

[예제 12-13] 구조체의 값에 의한 호출 코드

```c
#include <stdio.h>

struct human {                          // human 구조체 정의
   char name[10];                          // human 구조체 멤버: 이름(문자열)
   float power;                            // human 구조체 멤버: 힘(실수)
};                                      // human 구조체 정의 끝

void attack(struct human f1, struct human f2) {   // call by value
   if (f1.power > f2.power)
      printf("출동 %s\n", f1.name);
   else printf("출동 %s\n", f2.name);
}

int main() {
   struct human h1 = {"gildong", 8.2}, h2 = {"hyungwook", 9.3};

   attack(h1, h2);                                 // call by value

   return 0;
}
```

실행 화면

```
출동 hyungwook
```


12.5 구조체와 함수

[예제 12-14] 구조체의 참조에 의한 호출 코드

```c
#include <stdio.h>

struct human {                          // human 구조체 정의
   char name[10];                          // human 구조체 멤버: 이름(문자열)
   float power;                            // human 구조체 멤버: 힘(실수)
};                                      // human 구조체 정의 끝

void attack(struct human *f1, struct human *f2) { // call by reference
   if (f1->power > f2->power)
      printf("출동 %s\n", f1->name);
   else printf("출동 %s\n", f2->name);
}

int main() {
   struct human h1 = {"gildong", 8.2}, h2 = {"hyungwook", 9.3};

   attack(&h1, &h2);                               // call by reference

   return 0;
}
```

실행 화면

```
출동 hyungwook
```

[예제 12-15] 구조체의 값 반환방식 호출 코드

```c
#include <stdio.h>

struct human {                          // human 구조체 정의
    char name[10];                          // human 구조체 멤버: 이름(문자열)
    float power;                            // human 구조체 멤버: 힘(실수)
};                                      // human 구조체 정의 끝

struct human attack(struct human f1, struct human f2) {
    if (f1.power > f2.power)
        return f1;                                      // return value f1
    else return f2;                                     // return value f2
}

int main() {
    struct human h1 = {"gildong", 8.2}, h2 = {"hyungwook", 9.3}, att;

    att = attack(h1, h2);                               // return value
    printf("출동 %s\n", att.name);
    return 0;
}
```

실행 화면

```
출동 hyungwook
```


2 구조체 배열을 함수로 전달하기

[예제 12-16] 구조체 배열 출력 코드

```c
#include <stdio.h>

typedef struct human {              // typedef를 사용하여 human 구조체 정의
    char name[10];                      // human 구조체 멤버: 이름(문자열)
    int age;                            // human 구조체 멤버: 나이(정수)
    float power;                        // human 구조체 멤버: 힘(실수)
} human;                            // human 구조체 별명 human

void print_human(human *hp, int j) {
    int k;

    for (k = 0; k < j; k++)
        printf("%s %d %.1f\n", hp[k].name, hp[k].age, hp[k].power);
}

int main() {
    human ho[4] = { {"gildong", 22, 8.2}, {"hyungwook", 28, 9.3},
                    {"changsik", 21, 7.6}, {"hosik", 29, 8.1} };

    print_human(ho, 4);

    return 0;
}
```

실행 화면

```
gildong 22 8.2
hyungwook 28 9.3
changsik 21 7.6
hosik 29 8.1
```

12.6 공용체와 열거형

1 공용체

- 공용체는 구조체와 비슷하지만 멤버들이 하나의 공간을 공동으로 사용하는 점이 다름
- 공용체는 하나의 데이터에 대하여 서로 다르게 해석해야 할 일이 있거나 데이터의 일부만을 사용할 일이 있을 때 사용

공용체 정의 및 변수 선언 방법

```
union 공용체_이름 {
   자료형 변수명;
   자료형 변수명;
};

union 공용체_이름 변수이름;
```

공용체 정의 시작
멤버 1
멤버 1
공용체 정의 끝은 중괄호와 세미콜론!

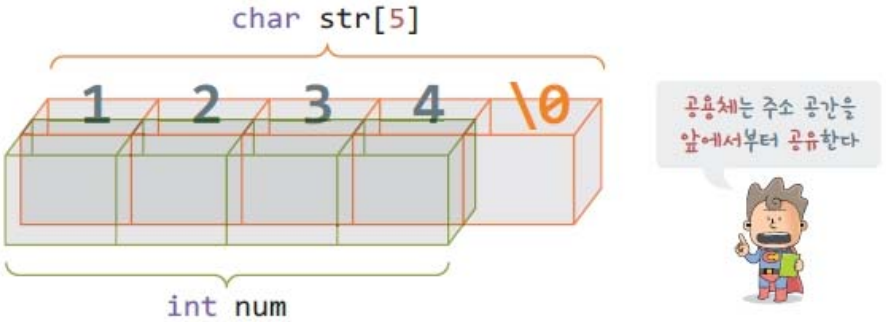
```
union cyto {
   char str[5];
   int num;
};

int main() {
   union cyto ct1;
```

공용체 cyto 정의
cyto 공용제 멤버: char str[5]: 5byte
cyto 구조체 멤버: int num: 4byte
str과 num 메모리 공유

공용체 cyto 변수 ct1 선언

12.6 공용체와 열거형

- 공용체 구조



[예제 12-17] 공용체 출력 코드

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

union cyto {                        // 공용체 cyto 정의
   char str[5];                       // cyto 공용제 멤버: char str[5]: 5byte
   int num;                           // cyto 구조체 멤버: int num: 4byte
};                                  // str과 num 메모리 공유

int main() {
   union cyto ct1;                  // 공용체 cyto 변수 ct1 선언

   strcpy(ct1.str, "1234");
   printf("정수:%d 문자열:%s\n", ct1.num, ct1.str);
   ct1.num = 1684234849;
   printf("정수:%d 문자열:%s\n", ct1.num, ct1.str);

   return 0;
}
```

실행 화면

```
정수:875770417 문자열:1234
정수:1684234849 문자열:abcd
```


[예제 12-18] 공용체를 이용한 암호변환 코드

```c
#include <stdio.h>

typedef union cyto {           // 공용체 cyto 선언
   char str[5];                   // cyto 공용제 멤버: char str[5]: 5byte
   short num[2];                  // cyto 구조체 멤버: short num[2]: 4byte
} cyto;                        // str과 num 메모리 공유

int main() {
   cyto ct1, ct2;

   printf("4글자 입력: ");
   scanf("%s", ct1.str);                        // ct1.str에 문자 4개 입력
   printf("암호는 %d %d\n\n", ct1.num[0], ct1.num[1]);

   printf("암호 숫자 2개 입력: ");
   scanf("%d %d", &ct2.num[0], &ct2.num[1]);  // ct2.num에 정수 2개 입력
   printf("문자는 %s\n", ct2.str);

   return 0;
}
```

실행 화면

```
4글자 입력: aqua
암호는 29025 24949

암호 숫자 2개 입력: 29025 24949
문자는 aqua
```

12.6 공용체와 열거형

2 열거형

- 열거형(enumeration)이란 변수가 가질 수 있는 값들을 모아 놓은 것
- 열거형은 enum 열거형_이름 { 멤버1, 멤버2 };로 정의
- 열거형의 멤버는 정수를 문자로 바꿔놓은 것 뿐

열거형 정의

```
enum week { SUN, MON, TUE, WED, THU, FRI, SAT };
```

```
enum week { SUN, MON, TUE, WED, THU, FRI, SAT };

int main() {
   enum week w1;

   w1 = MON;
   printf("%d\n", w1);
```

```
열거형 변수 w1 선언;

w1에 1이 들어감;
숫자 1출력
```


12.6 공용체와 열거형

- 열거형은 다음과 같이 멤버를 #define을 사용하여 상수로 정의하는 것과 같음
- 열거형은 기본적으로 0부터 시작하는 정수가 차례로 배정
- 만약 멤버에 배정되는 값을 1부터 시작하고 싶다면, 멤버 맨 앞의 값에게 SUN = 1과 같이 시작 값을 주면 됨

```
#define SUN 0
#define MON 1
#define TUE 2
.
.
```

```
enum week { SUN = 1, MON, TUE, WED, THU, FRI, SAT };
```

```
enum week { RED = 7, GREEN = 10, BLUE = 13};
```

```
enum { FALSE, TRUE };
```

[예제 12-19] 열거형을 이용한 소개팅 코드

```c
#include <stdio.h>

enum { FALSE, TRUE };                          // 열거형 선언

int main() {
   int love;

   printf("이거를 마신다? YES=1, NO=0: ");
   scanf("%d", &love);
   if (love == TRUE)                           // 불린 변수 사용가능
       printf("둘이 사귄다\n");
   else
       printf("헤어진다\n");

   return 0;
}
```

실행 화면

```
이거를 마신다? YES=1, NO=0: 1
둘이 사귄다
```


09 다음 중 구조체와 똑같지만 멤버들이 하나의 공간을 공동으로 사용하는 자료형을 정의 할 때 사용하는 단어는 무엇인가?

① struct ② enum ③ typedef ④ union

10 다음 중 변수가 가질 수 있는 값들을 모아 놓은 것이라는 뜻을 가진 단어는 무엇인가?

① struct ② enum ③ typedef ④ union

11 다음 중 자료형의 별명을 만들 때 사용하는 단어는 무엇인가?

① struct ② enum ③ typedef ④ union

12 h1과 h2가 구조체인 경우 허용되는 연산은 무엇인가?

① h1 == h2 ② h1 < h2 ③ h1 > h2 ④ h1 = h2

13 h1과 h2가 구조체인 경우 h2를 h1에 복사하는 연산은 무엇인가?

① h1 == h2 ② h1 < h2 ③ h1 > h2 ④ h1 = h2

14 다음 중 점을 사용하지 않고 포인터를 이용하여 구조체 멤버에 접근할 때에 사용하는 기호는 무엇인가?

① 화살표(->) ② 느낌표(!) ③ 별(*) ④ 쉼표(,)

15 다음과 같이 자신을 가리키는 구조체를 정의하려 한다. 다음 괄호에 들어갈 적당한 단어는 무엇인가?

```
[          ] struct human {
char name[10];
struct human *next;
} human;
```

① struct ② enum ③ typedef ④ union

16 다음과 같이 자신을 가리키는 구조체를 정의하려 한다. 다음 괄호에 들어갈 적당한 단어는 무엇인가?

```
typedef struct human {
char name[10];
[          ] human *next;
} human;
```

① struct ② enum ③ typedef ④ union

26 구조체와 똑같지만 멤버들이 하나의 공간을 공동으로 사용하는 자료형을 정의할 때 사용하는 단어는 ( )이다.

27 변수가 가질 수 있는 값들을 모아 놓은 것이라는 뜻을 가진 단어는 ( )이다.

28 자료형의 별명을 만들 때 사용하는 단어는 ( )이다.

30 h1과 h2가 구조체인 경우 h2를 h1에 복사하는 연산은 ( )이다.

31 점을 사용하지 않고 포인터를 이용하여 구조체 멤버에 접근할 때에는 기호는 ( )이다.

32 다음 코드는 자기 자신을 가리키는 구조체를 선언한 것이다. 괄호 안에 들어갈 단어는 ()이다.

```
                    struct human {
  char name[10];
  struct human *next;
  } human;
```


33 다음 코드는 자기 자신을 가리키는 구조체를 선언한 것이다. 괄호 안에 들어갈 단어는 ( )이다.

```
typedef struct human {
    char name[10];
                    human *next;
} human;
```