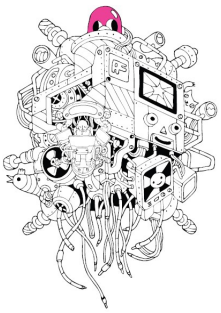


Chapter 23. 구조체 part 2

구조체의 정의와 typedef 선언

typedef 선언

```
typedef  int  INT;
```

자료형의 이름 int에 INT라는 이름을 추가로 붙여줍니다.

위의 typedef 선언으로 인해서!!!

```
INT num;
INT * ptr;
```

int num; 과 동일한 선언

int * ptr; 과 동일한 선언

새로 부여된 이름	대상 자료형
INT	int
PTR_INT	int *
UINT	unsigned int
PTR_UINT	unsigned int *
UCHAR	unsigned char
PTR_UCHAR	unsigned char *

정의되는 이름들

TypeNameTypedef.c

```
typedef int INT;
typedef int * PTR_INT;

typedef unsigned int UINT;
typedef unsigned int * PTR_UINT;

typedef unsigned char UCHAR;
typedef unsigned char * PTR_UCHAR;

int main(void)
{
    INT num1 = 120;             // int num1 = 120;
    PTR_INT pnum1 = &num1;      // int * pnum1 = &num1;

    UINT num2 = 190;            // unsigned int num2 = 190;
    PTR_UINT pnum2 = &num2;     // unsigned int * pnum2 = &num2;

    UCHAR ch = 'Z';             // unsigned char ch = 'Z';
    PTR_UCHAR pch = &ch;        // unsigned char * pch = &ch;

    printf("%d, %u, %c \n", *pnum1, *pnum2, *pch);
    return 0;
}
```

실행결과

```
120, 190, Z
```

3

구조체 정의와 typedef 선언

```
struct point
{
    int xpos;
    int ypos;
};

typedef struct point Point;
```

구조체 point 정의 후
struct point에 Point라는 이름을 부여하기 위한 typedef 선언 추가!

합친 형태

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;
```

구조체 point의 정의와
Point에 대한 typedef 선언을 한데 묶은 형태

4

typedef 선언

- Examples
  - typedef int INT4;
  - typedef short INT2;
  - typedef char INT1;
  - typedef float REAL4;
  - typedef double REAL8;
  - INT4 a = 10; ← → int a = 10;
  - REAL4 b = 20.4; ← → float b = 20.4;
  - REAL8 c = 378.45; ← → double c = 378.45

5

구조체 정의와 typedef 선언 관련 예제

StructTypedef.c

```
struct point
{
    int xpos;
    int ypos;
};

typedef struct point Point;
```

구조체 point의 정의와 typedef 선언

```
typedef struct person
{
    char name[20];
    char phoneNum[20];
    int age;
} Person;
```

구조체 person의 정의와
Person이라는 이름의 typedef 선언을 하나로!

```
int main(void)
{
    Point pos={10, 20};
    Person man={"이승기", "010-1212-0001", 21};
    printf("%d %d \n", pos.xpos, pos.ypos);
    printf("%s %s %d \n", man.name, man.phoneNum, man.age);
    return 0;
}
```

실행결과

```
10 20
이승기 010-1212-0001 21
```

6

구조체의 이름 생략

```
typedef struct person
{
    char name[20];
    char phoneNum[20];
    int age;
} Person;
```

typedef 선언으로 인해서 새로운 이름 Person이 정의 되었으니, 구조체의 이름 person은 큰 의미가 없다.

이름이 생략된 형태

```
typedef struct
{
    char name[20];
    char phoneNum[20];
    int age;
} Person;
```

struct person man; // 불가능한 선언

Person은 변수의 이름이 아니고 자료형의 이름이다.

7

복소수 구조체

- 복소수를 표현하는 구조체 complex를 정의 (예제)
  - 복소수는 a+bi로 표현되며, 실수부 a와 허수부 b는 실수 값
  - 구조체 struct complex를 다음과 같이 정의

```
struct complex {
        double real; //실수부
        double img; //허수부
};

typedef struct complex Complex;
Complex comp1, comp2;
```

8

프로그램 예제

- 두 복소수를 더하는 프로그램

```
#include <stdio.h>
struct complex {
            double real;  //실수부
            double img;  //허수부
};
typedef struct complex Complex;

int main(void) {
    Complex comp1 = {3, 4}, comp2 = {2, 5}, comp3; // 3+4i, 2+5i

    comp3.real = comp1.real + comp2.real;
    comp3.img = comp1.img + comp2.img;

    printf("(%.2lf+%.2lfi) + (%.2lf+%.2lfi) = %.2lf+%.2lfi\n", \
           comp1.real, comp1.img, comp2.real, comp2.img, \
           comp3.real, comp3.img);
    return 0;
}
```

실행결과

```
(3.00+4.00i) + (2.00+5.00i) = 5.00+9.00i
```

9

연습 문제 (연습1)

- 학생의 이름, 학번, 학과, 학년을 한꺼번에 저장하기 위해 구조체 struct student를 정의
- 이것을 typedef를 이용하여 struct student와 동일한 type인 Student를 정의
- main 함수에서 학생들의 정보를 저장하기 위해 Student type의 배열을 선언 (방의 개수는 5)
- 2명의 정보를 입력 받아 배열에 저장
- 2명의 정보를 main 함수 내에서 출력하기

입출력 형태

```
이름은? 홍길동
학번은? 20110000
학과는? 정보통신공학과
학년은? 1
이름은? 강채윤
학번은? 20119999
학과는? 정보통신공학과
학년은? 1
첫 번째 학생은
이름 : 홍길동
학번 : 20110000
학과 : 정보통신공학과
학년 : 1
두 번째 학생은
이름 : 강채윤
학번 : 20119999
학과 : 정보통신공학과
학년 : 1
```

10

함수로의 구조체 변수 전달과 변환

함수의 인자로 전달되고 return문에 의해 반환되는 구조체 변수1

StructValAndFunction.c

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;

void ShowPosition(Point pos)
{
    printf("[%d, %d] \n", pos.xpos, pos.ypos);
}

Point GetCurrentPosition(void)
{
    Point cen;
    printf("Input current pos: ");
    scanf("%d %d", &cen.xpos, &cen.ypos);
    return cen;
}

int main(void)
{
    Point curPos=GetCurrentPosition();
    ShowPosition(curPos);
    return 0;
}
```

구조체 변수 cen이 통째로 반환된다.

ShowPosition 함수의 매개변수에 curPos에 저장된 값이 통째로 복사된다.

실행결과

```
Input current pos: 2 4
[2, 4]
```

배열까지도 통째로 복사

StructMemArrCopy.c

```
typedef struct person
{
    char name[20];
    char phoneNum[20];
    int age;
} Person;
```

구조체의 멤버로 배열이 선언된 경우
구조체 변수를 인자로 전달하거나 반환 시
배열까지도 통째로 복사가 이뤄진다.

```
void ShowPersonInfo(Person man)
{
    printf("name: %s \n", man.name);
    printf("phone: %s \n", man.phoneNum);
    printf("age: %d \n", man.age);
}
Person ReadPersonInfo(void)
{
    Person man;
    printf("name? "); scanf("%s", man.name);
    printf("phone? "); scanf("%s", man.phoneNum);
    printf("age? "); scanf("%d", &man.age);
    return man;
}
int main(void)
{
    Person man=ReadPersonInfo();
    ShowPersonInfo(man);
    return 0;
}
```

실행결과

```
name? Jung
phone? 010-12XX-34XX
age? 22
name: Jung
phone: 010-12XX-34XX
age: 22
```

구조체 기반의 Call-by-reference

StructFunctionCallByRef.c

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;
void OrgSymTrans(Point * ptr)    // 원점대칭
{
    ptr->xpos = (ptr->xpos) * -1;
    ptr->ypos = (ptr->ypos) * -1;
}
void ShowPosition(Point pos)
{
    printf("[%d, %d] \n", pos.xpos, pos.ypos);
}
int main(void)
{
    Point pos={7, -5};
    OrgSymTrans(&pos);    // pos의 값을 원점 대칭이동시킨다.
    ShowPosition(pos);
    OrgSymTrans(&pos);    // pos의 값을 원점 대칭이동시킨다.
    ShowPosition(pos);
    return 0;
}
```

구조체 변수 대상의 Call-by-reference는
일반 변수의 Call-by-reference와 동일하다.

실행결과

```
[-7, 5]
[7, -5]
```

구조체 변수를 대상으로 가능한 연산1

StructOperation.c

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;

int main(void)
{
    Point pos1={1, 2};
    Point pos2;
    pos2=pos1;  // pos1의 멤버 대 pos2의 멤버간 복사가 진행됨

    printf("크기: %d \n", sizeof(pos1));   // pos1의 전체 크기 반환
    printf("[%d, %d] \n", pos1.xpos, pos1.ypos);
    printf("크기: %d \n", sizeof(pos2));   // pos2의 전체 크기 반환
    printf("[%d, %d] \n", pos2.xpos, pos2.ypos);
    return 0;
}
```

구조체 변수간 대입연산의 결과로 **멤버 대 멤버 복사**가 이뤄진다는 사실을 확인하자!

실행결과

```
크기: 8
[1, 2]
크기: 8
[1, 2]
```

구조체 변수를 대상으로 가능한 연산2

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;
```

구조체 변수를 대상으로는 덧셈 및 뺄셈 연산이 불가능하다.
따라서 필요하다면 덧셈함수와 뺄셈함수를 정의해야 한다.

```
Point AddPoint(Point pos1, Point pos2)
{
    Point pos={pos1.xpos+pos2.xpos, pos1.ypos+pos2.ypos};
    return pos;
}
```

구조체 Point의 덧셈 함수

```
Point MinPoint(Point pos1, Point pos2)
{
    Point pos={pos1.xpos-pos2.xpos, pos1.ypos-pos2.ypos};
    return pos;
}
```

구조체 Point의 뺄셈 함수

```
int main(void)
{
    Point pos1={5, 6};
    Point pos2={2, 9};
    Point result;

    result=AddPoint(pos1, pos2);
    printf("[%d, %d] \n", result.xpos, result.ypos);
    result=MinPoint(pos1, pos2);
    printf("[%d, %d] \n", result.xpos, result.ypos);
    return 0;
}
```

StructAddMin.c

실행결과

```
[7, 15]
[3, -3]
```

연습 문제 (연습2)

- 학생의 이름, 학번, 학과, 학년을 한꺼번에 저장하기 위해 구조체 struct student를 정의
- 이것을 typedef를 이용하여 struct student와 동일한 type인 Student를 정의
- main 함수에서 학생들의 정보를 저장하기 위해 Student type의 배열을 선언 (방의 개수는 5)
- 2명의 정보를 입력 받아 배열에 저장

- 각 학생의 정보를 차례로 PrintStudent라는 함수로 보내어 PrintStudent 함수에서 학생의 이름, 학번, 학과, 학년을 출력
  - void PrintStudent(Student * stu); // 함수선언
  - 구조체 변수의 포인터에서 사용하는 ->를 사용하여 출력

입출력 형태

```
이름은? 홍길동
학번은? 20110000
학과는? 정보통신공학과
학년은? 1
이름은? 강채윤
학번은? 20119999
학과는? 정보통신공학과
학년은? 1
첫 번째 학생은
이름 : 홍길동
학번 : 20110000
학과 : 정보통신공학과
학년 : 1
두 번째 학생은
이름 : 강채윤
학번 : 20119999
학과 : 정보통신공학과
학년 : 1
```

17

구조체의 유용함 및 중첩 구조체

구조체를 정의하는 이유

- 연관 있는 데이터를 하나로 묶을 수 있는 자료형을 정의할 수 있다.
- 연관 있는 데이터를 묶으면 데이터의 표현 및 관리가 용이해진다.
- 데이터의 표현 및 관리가 용이해지면 그만큼 합리적인 코드를 작성할 수 있다.

구조체의 정의 이유!

```
typedef struct student
{
    char name[20];      // 학생 이름
    char stdnum[20];    // 학생 고유번호
    char school[20];    // 학교 이름
    char major[20];     // 선택 전공
    int year;           // 학년
} Student;
```

struct student와 Student는 동일

```
void ShowStudentInfo(Student * sptr)
{
    printf("학생 이름: %s \n", sptr->name);
    printf("학생 고유번호: %s \n", sptr->stdnum);
    printf("학교 이름: %s \n", sptr->school);
    printf("선택 전공: %s \n", sptr->major);
    printf("학년: %d \n", sptr->year);
}
```

인자 전달 시 용이

```
int main(void)
{
    Student arr[7];
    int i;

    for(i=0; i<7; i++)
    {
        printf("이름: "); scanf("%s", arr[i].name);
        printf("번호: "); scanf("%s", arr[i].stdnum);
        printf("학교: "); scanf("%s", arr[i].school);
        printf("전공: "); scanf("%s", arr[i].major);
        printf("학년: "); scanf("%d", &arr[i].year);
    }

    for(i=0; i<7; i++)
        ShowStudentInfo(&arr[i]);
    return 0;
}
```

하나의 배열 선언으로 종류가 다른 데이터들을 한데 저장할 수 있다.

StructImportant.c

19

중첩된 구조체의 정의와 변수의 선언

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;

typedef struct circle
{
    Point cen;
    double rad;
} Circle;
```

```
void ShowCircleInfo(Circle * cptr)
{
    printf("[%d, %d] \n", (cptr->cen).xpos, (cptr->cen).ypos);
    printf("radius: %g \n\n", cptr->rad);
}

int main(void)
{
    Circle c1={{1, 2}, 3.5};
    Circle c2={2, 4, 3.9};
    ShowCircleInfo(&c1);
    ShowCircleInfo(&c2);
    return 0;
}
```

CircleIncludePoint.c

앞서 정의한 구조체는 이후에 새로운 구조체를 선언하는데 있어서 기본 자료형의 이름과 마찬가지로 사용이 될 수 있다.

실행결과

```
[1, 2]
radius: 3.5

[2, 4]
radius: 3.9
```

20

공용체(Union Type)의 정의와 의미

구조체 vs. 공용체: 선언방식의 차이

```
typedef struct sbox
{
    int mem1;
    int mem2;
    double mem3;
} SBox;
```

```
typedef union ubox
{
    int mem1;
    int mem2;
    double mem3;
} UBox;
```

정의 방법에 있어서의 차이는 키워드 struct를 쓰느냐, 아니면 키워드 union을 쓰느냐에 있다!

22

구조체 vs. 공용체: 실행결과를 통한 관찰

공용체 변수를 이루는 멤버의 시작 주소 값이 모두 동일함을 관찰하고 공용체 변수의 크기 값을 관찰한다!

```
typedef struct sbox
{
    int mem1;
    int mem2;
    double mem3;
} SBox;

typedef union ubox
{
    int mem1;
    int mem2;
    double mem3;
} UBox;
```

실행결과

```
002CFC28 002CFC2C 002CFC30
002CFC18 002CFC18 002CFC18
16 8
```

```
int main(void)
{
    SBox sbx;
    UBox ubx;
    printf("%p %p %p \n", &sbx.mem1, &sbx.mem2, &sbx.mem3);
    printf("%p %p %p \n", &ubx.mem1, &ubx.mem2, &ubx.mem3);
    printf("%d %d \n", sizeof(SBox), sizeof(UBox));
    return 0;
}
```

UnionMemAlloc.c

23

구조체 vs. 공용체: 메모리적 차이

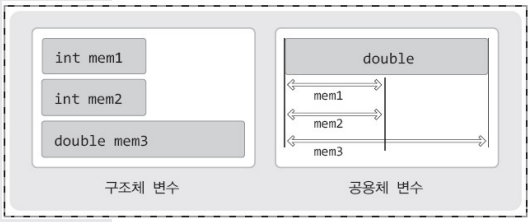
```
typedef union ubox    // 공용체 ubox의 정의
{
    int mem1;
    int mem2;
    double mem3;
} UBox;

int main(void)
{
    UBox ubx;    // 8바이트 메모리 할당
    ubx.mem1=20;
    printf("%d \n", ubx.mem2);
mem1에 저장된 데이터를 덮어쓴다.
    ubx.mem3=7.15;
    printf("%d \n", ubx.mem1);
    printf("%d \n", ubx.mem2);
    printf("%g \n", ubx.mem3);
    return 0;
}
```

UnionValAccess.c

```
20
-1717986918
-1717986918
7.15
```

실행결과



24

공용체의 유용함1: 문제의 제시

- 민선: 수진아! 교수님이 과제를 내 주셨어
- 수진: 뭔데?
- 민선: 프로그램 사용자로부터 int형 정수 하나를 입력 받으래
- 수진: 그래서?
- 민선: 입력 받은 정수의 상위 2바이트와 하위 2바이트 값을 양의 정수로 출력!
- 수진: 그게 다야?
- 민선: 그 다음엔 상위 1바이트와 하위 1바이트에 저장된 값의 아스키 문자 출력!
- 수진: 그거 공용체를 이용해 보라는 깊은 뜻이 담겨있는 것 같은데?

```
typedef struct dbshort
{
    unsigned short upper;
    unsigned short lower;
} DBShort;

typedef union rdbuf
{
    int iBuf;
    char bBuf[4];
    DBShort sBuf;
} RDBuf;
```

해결책이 되는 공용체의 정의

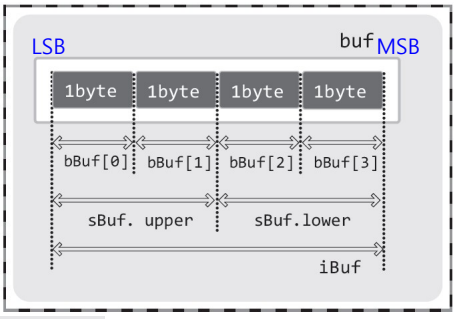
공용체의 유용함 → 다양한 접근방식을 제공하는 것!

25

공용체의 유용함2: 문제의 해결

```
typedef struct dbshort
{
    unsigned short upper;
    unsigned short lower;
} DBShort;
typedef union rdbuf
{
    int iBuf;
    char bBuf[4];
    DBShort sBuf;
} RDBuf;
int main(void)
{
    RDBuf buf;
    printf("정수 입력: ");
    scanf("%d", &(buf.iBuf));
    printf("상위 2바이트: %u \n", buf.sBuf.upper);
    printf("하위 2바이트: %u \n", buf.sBuf.lower);
    printf("상위 1바이트 아스키 코드: %c \n", buf.bBuf[0]);
    printf("하위 1바이트 아스키 코드: %c \n", buf.bBuf[3]);
    return 0;
}
```

UsefulUnionAccess.c



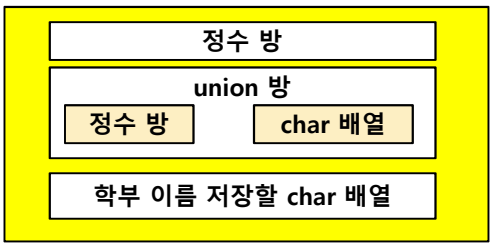
실행결과

```
정수 입력: 1145258561
상위 2바이트: 16961
하위 2바이트: 17475
상위 1바이트 아스키 코드: A
하위 1바이트 아스키 코드: D
```

26

실습문제 (Lab 2)

- 학생 정보를 한꺼번에 저장하기 위해 구조체 struct student를 정의
  - 학번과 이름 중 어느 것을 입력하는지를 저장한 int 방 1개
  - 학번 또는 이름 저장할 union idORname type의 방 1개
  - 학부 명을 저장한 char 배열 1개
  - union idORname에는 학번을 저장할 int 방 1개, 이름을 저장할 char 배열 1개로 정의할 것
- 최종 방의 모양이 아래와 같도록 하면 된다.



27

실습문제 (Lab 2: cont'd)

- typedef를 이용, struct student와 동일한 type인 Student를 정의
- main 함수에서 Student type의 배열을 선언 (방의 개수는 5)
- 각 학생의 정보를 차례로 PrintStudent라는 함수로 보내어 학생 정보 출력. (PrintStudent 함수를 두 번 호출!)
  - void PrintStudent(Student * stu); ←→ void PrintStudent(Student stu) 차이점???
  - 구조체 변수의 포인터에서 사용하는 ->를 사용하여 출력

```
학생의 학번과 이름 중 어느 것을 입력합니까? (1: 학번, 2: 이름) : 1
학번을 입력하세요 : 20141111
학과 또는 학부를 입력하세요 : 정보통신공학과

학생의 학번과 이름 중 어느 것을 입력합니까? (1: 학번, 2: 이름) : 2
이름을 입력하세요 : 홍길동
학과 또는 학부를 입력하세요 : 전자IT기계자동차공학부

(1). 20141111 학생은 정보통신공학과입니다.
(2). 홍길동 학생은 전자IT기계자동차공학부입니다.
```

28

열거형(Enumerated Type)의 정의와 의미

열거형의 정의와 변수의 선언

▶ 열거형 syllable의 정의의 의미

"syllable형 변수에 저장이 가능한 정수 값들을 결정하겠다"

▶ 열거형 syllable의 정의의 예

```
enum syllable    // syllable이라는 이름의 열거형 정의
{
    Do=1, Re=2, Mi=3, Fa=4, So=5, La=6, Ti=7
};
```

Do, Re, Mi, Fa. . . .
를 열거형 상수라 한다.

Do를 정수 1을 의미하는 상수로 정의한다.

그리고 이 값은 syllable형 변수에 저장이 가능하다

▶ syllable형 변수의 선언

enum syllable tone; // syllable형 변수 tone의 선언

구조체 공용체와 마찬가지로 typedef 선언을 추가하여 enum 선언을 생략할 수 있다.

30

열거형 이용

- enum의 활용 예제
  - enum day7을 정의하고, typedef를 이용하여 열거형 자료형 day를 정의

```
enum day7 {sun, mon, tue, wed, thu, fri, sat};
typedef  enum  day7  day;
```

  - 새로운 자료형 day는 enum day7과 동일
  - 변수를 선언하면서 초기 값으로 상수 fri를 대입하는 문장

```
day today = fri; // enum day7 today = fri;와 동일
```

31

열거형의 정의와 변수선언의 예

```
typedef enum syllable
{
    Do=1, Re=2, Mi=3, Fa=4, So=5, La=6, Ti=7
} Syllable;
```

typedef 선언이 추가된 열거형의 정의 및 선언

실행결과

```
도는 하얀 도라지 ♪
레는 둥근 레코드 ♩
미는 파란 미나리 ♩♪
파는 예쁜 파랑새 ♪♭
솔은 작은 솔방울 ♩♪♪
라는 라디오고요~ ♪♩♭♩
시는 졸졸 시냇물 ♩♭♩♪
```

열거형 상수는 선언 이후 어디서건 쓸 수 있는 상수가 된다.

EnumTypeTone.c

```
void Sound(Syllable sy)
{
    switch(sy)
    {
    case Do:
        puts("도는 하얀 도라지 ♪"); return;
    case Re:
        puts("레는 둥근 레코드 ♩"); return;
    case Mi:
        puts("미는 파란 미나리 ♩♪"); return;
    case Fa:
        puts("파는 예쁜 파랑새 ♪♭"); return;
    case So:
        puts("솔은 작은 솔방울 ♩♪♪"); return;
    case La:
        puts("라는 라디오고요~ ♪♩♭♩"); return;
    case Ti:
        puts("시는 졸졸 시냇물 ♩♭♩♪"); return;
    }
    puts("다 함께 부르세~ 도레미파 솔라시도 솔 도~ 짠~");
}
int main(void)
{
    Syllable tone;
    for(tone=Do; tone<=Ti; tone+=1)
        Sound(tone);
    return 0;
}
```

32

열거형 상수의 값이 결정되는 방식

```
enum color {RED, BLUE, WHITE, BLACK};
```

동일한 선언

```
enum color {RED=0, BLUE=1, WHITE=2, BLACK=3};
```

열거형 상수의 값은 명시되지 않으면 0부터 시작해서 1씩 증가한다.

```
enum color {RED=3, BLUE, WHITE=6, BLACK};
```

동일한 선언

```
enum color {RED=3, BLUE=4, WHITE=6, BLACK=7};
```

값이 명시되지 않는 상수는 앞에 정의된 상수 값에서 1이 증가한다.

33

problem

- 다음 문장에서 잘못된 것은 무엇인가?

```
enum carmaker {kia, hyundai, samsung, Daewoo, ford};
carmaker = Hyundai;
```

- 다음 문장의 출력값은 무엇인가?

```
enum day7 {sun, mon, tue, wed, thu, fri, sat};
typedef enum day7 day;
day today = sat;
printf("today = %d₩n", today);
```

34

solution

- 다음 문장에서 잘못된 것은 무엇인가?

```
enum carmaker {kia, hyundai, samsung, Daewoo, ford};
carmaker = Hyundai;
```

  - 위 문장에서 식별자 carmaker는 열거형의 태그이지 변수가 아니다. 그러므로 변수를 하나 선언하여 적당한 값을 대입해야 한다.그리고 대입할 수 있는 값은 Hyundai가 아니라 hyundai이다.

```
enum carmaker car;
car = hyundai;
```

- 다음 문장의 출력값은 무엇인가?

```
enum day7 {sun, mon, tue, wed, thu, fri, sat};
typedef enum day7 day;
day today = sat;
printf("today = %d₩n", today);
```

  - 열거형에서 정수 상수를 특별히 지정하지 않으면 0에서부터 시작하는 정수이므로 6이 출력된다.

35

열거형의 유용함

```
typedef enum syllable
{
    Do=1, Re=2, Mi=3, Fa=4, So=5, La=6, Ti=7
} Syllable;
```

Syllable이라는 이름의 자료형 안에서 음계에 관련있는 상수들을 모두 묶어서 정의하였다!

```
enum {
  Do=1, Re=2, Mi=3, Fa=4, So=5, La=6, Ti=7 };
```

변수의 선언이 목적이 아닌 상수의 선언이 목적인 경우 이렇듯 열거형의 이름과 typedef 선언을 생략하기도 한다.

열거형의 유용함은 둘 이상의 연관이 있는 이름을 상수로 선언함으로써 프로그램의 가독성을 높이는데 있다.

36

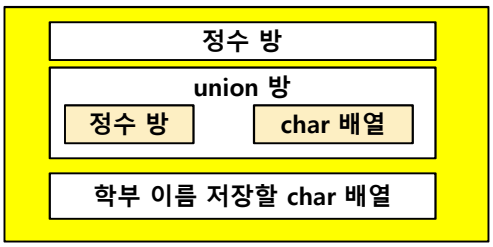
struct, union, enum

- 모두다 변수의 정의가 아닌 자료형의 정의
  - struct student { .... };
  - union ubox { .... };
  - enum day7 { .... };
- 변수 정의 방법도 동일
  - struct student human; // human이라는 방
  - union ubox ubx; // ubx라는 방
  - enum day7 day; // day라는 방

37

실습문제 (Lab 2)

- 학생 정보를 한꺼번에 저장하기 위해 구조체 struct student를 정의
  - 학번과 이름 중 어느 것을 입력하는지를 저장한 int 방 1개
  - 학번 또는 이름 저장할 union idORname type의 방 1개
  - 학부 명을 저장한 char 배열 1개
  - union idORname에는 학번을 저장할 int 방 1개, 이름을 저장할 char 배열 1개로 정의할 것
- 최종 방의 모양이 아래와 같도록 하면 된다.



38

실습문제 (Lab 2: cont'd)

- typedef를 이용, struct student와 동일한 type인 Student를 정의
- main 함수에서 Student type의 배열을 선언 (방의 개수는 5)
- 각 학생의 정보를 차례로 PrintStudent라는 함수로 보내어 학생 정보 출력. (PrintStudent 함수를 두 번 호출!)
  - void PrintStudent(Student * stu); ←→ void PrintStudent(Student stu) 차이점???
  - 구조체 변수의 포인터에서 사용하는 ->를 사용하여 출력

```
학생의 학번과 이름 중 어느 것을 입력합니까? (1: 학번, 2: 이름) : 1
학번을 입력하세요 : 20141111
학과 또는 학부를 입력하세요 : 정보통신공학과

학생의 학번과 이름 중 어느 것을 입력합니까? (1: 학번, 2: 이름) : 2
이름을 입력하세요 : 홍길동
학과 또는 학부를 입력하세요 : 전자IT기계자동차공학부

(1). 20141111 학생은 정보통신공학과입니다.
(2). 홍길동 학생은 전자IT기계자동차공학부입니다.
```

39

실습 시간 (2019년 11월 12일)

- 예제 (23장): CircleincludePoint.c, EnumTypeTone.c StructAddMin.c, StructFunctionCallByRef.c, StructImportant.c, StructMemArrCopy.c, StructOperation.c, StructTyedef.c, ~~(StructValAndFunction.c, TypeNameTypedef.c), (UnionMemAlloc.c),~~ UnionValAccess.c, UsefulUnionAccess.c (10개)

- Lab 문제: Lab 2

Chapter 23이 끝났습니다. 질문 있으신지요?