8장 구조체 공용체

한림대학교 소프트웨어융합대학 양은샘.

8장 구조체 공용체

- 안녕하세요? 여러분!

- 이번 장에서는 구조체와 공용체에 대해 학습합니다.

- C 언어에서 구조체는 객체지향언어의 클래스에 해당하는 기능을 합니다. 공용체는 구조체의 특별한 형태입니다.
- C 언어는 구조체를 사용하여 객체지향언어의 클래스처럼 다양한 자료형의 데이터를 묶음으로 사용합니다.

- 지난 시간에 학습한 내용을 리뷰한 후 학습을 시작하도록 하겠습니다.

지난 시간 Review

7.1 함수의 정의
7.2 표준 라이브러리 함수
7.3 함수의 호출 및 복귀
7.4 매개 변수의 전달 방식
7.5 함수와 변수영역
7.6 순환 호출(Recursive Call)
7.7 함수와 배열
7.8 main() 함수의 매개변수
7.9 함수 포인터
7.10 배열 포인터 함수 응용
□ 개념 확인 학습
□ 적용 확인 학습
□ 응용 프로그래밍

8장 구조체 공용체

8.1 구조체 정의와 구조체 변수의 선언

8.2 구조체 변수의 배열과 포인터

8.3 중첩된 구조체

8.4 자기 참조 구조체

8.5 구조체와 함수

8.6 비트 필드 구조체

8.7 공용체

8.8 구조체와 공용체의 응용

□ 개념 확인 학습

□ 적용 확인 학습

□ 응용 프로그래밍

학습 목표

- 구조체의 정의를 이해한다.
- 구조체를 사용하여 다양한 자료형을 묶음으로 사용할 수 있다.
- 구조체 포인터와 구조체 배열을 이용할 수 있다.
- 함수에서 구조체 포인터를 이용할 수 있다.
- 공용체의 의미를 이해하고 사용할 수 있다.
- 구조체 배열과 포인터를 응용하여 다양한 조건의 문제들을 해결 할 수 있다.

- 개념 확인 학습으로 배운 내용을 정리한다.
- 적용 확인 학습으로 개념 습득 여부를 확인한다.
- 응용 프로그래밍으로 문제해결력을 키운다.

구조체 정의

- 구조체 정의란?
  - 하나의 새로운 자료형을 만드는 과정
  - struct 키워드를 사용하여 구조체의 명칭을 정하고 필요한 멤버 변수들을 포함해 주는 것을 의미

  - 예)
    - "누구누구의 몇 번째 생일을 축하합니다."라는 문자를 보내주는 시스템을 만든다고 가정한다면,
    - 필요한 정보는 이름, 나이, 생년월일, 전화번호.
    - 이 정보들을 하나로 묶어 구조체를 만들어 사용한다면 데이터를 관리하기가 편리 함.

구조체 정의 형식

```
struct 구조체명 {   //키워드 struct 사용
    멤버변수 1;
     ……
    멤버변수 n;
}; //세미콜론 필요
```

구조체 정의 예

```
struct neighbor { //struct neighbor 자료형 정의
  char name[13];
  int age;
  char phone[13];
};
```

구조체 변수의 선언

구조체 변수 선언 형식

struct 구조체명 구조체변수명;

구조체 변수 선언 예

1) 구조체 정의 단계에서 구조체 변수 선언

```
struct neighbor { //struct neighbor자료형의
    char name[13];
    int age;
    char phone[13];
} his; //구조체 변수 his 선언
```

2) 구조체 정의 후 구조체 변수 선언

```
struct neighbor { //struct neighbor자료형의
    char name[13];
    int age;
    char phone[13];
};
struct neighbor his; //구조체 변수 his 선언
```

3) 구조체 정의 후 구조체명 재정의 및 구조체 변수 선언

```
struct neighbor { //struct neighbor정의
    char name[13];
    int age;
    char phone[13];
};
//typedef로 struct neighbor 자료형의 명칭을 neighbor로 재정의
typedef struct neighbor neighbor;
neighbor his; //구조체 변수 his 선언
```

4) typedef로 구조체 정의 후 구조체 변수 선언

```
typedef struct { //typedef로 구조체 정의와 동시에
    char name[13];
    int age;
    char phone[13];
} neighbor; //구조체명칭을 neighbor로 지정
neighbor his; //구조체 변수 his 선언
```

구조체 변수의 초기화와 참조

구조체 정의 단계에서 선언된 구조체 변수의 초기화

```
struct neighbor {
        char name[13];
        int age;
        char phone[13];
} her = {"raina", 25, "010-111-2222"};
```

구조체 정의 및 재정의 후 선언된 구조체 변수의 초기화

```
struct neighbor {
        char name[13];
        int age;
        char phone[13];
};
typedef struct neighbor neighbor;
neighbor her = {"raina", 25, "010-111-2222"};
```

구조체변수명.멤버변수로 접근

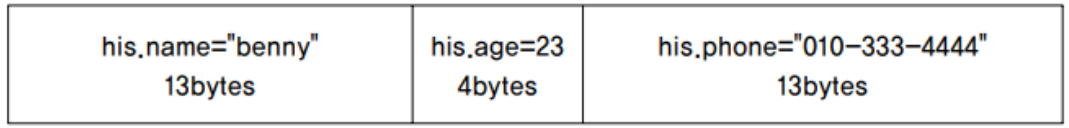
```
printf("her.name=%s, her.age=%d", her.name, her.age);
```

구조체 멤버 변수의 참조

[예제 8.1] 구조체 멤버 변수의 참조

```
#include <stdio.h>

struct neighbor {
        char name[13];
        int age;
        char phone[13];
};
typedef struct neighbor neighbor;

int main(int argc, char *argv[])
{
        neighbor his;

        printf("sizeof(his.name) = %d bytes₩n", sizeof(his.name));
        printf("sizeof(his.age) = %d bytes₩n", sizeof(his.age));
        printf("sizeof(his.phone) = %d bytes₩n", sizeof(his.phone));
        printf("sizeof(his) = %d bytes₩n", sizeof(his));

        printf("₩n이름 : ");
        gets(his.name);

        printf("나이 : ");
        scanf("%d", &his.age);

        while (getchar() != '₩n'); //버퍼에 저장되어 있는 값 삭제

        printf("전화번호 : ");
        gets(his.phone);

        printf("his.name=%s, his.age=%d, his.phone=%s₩n", his.name, his.age,
his.phone);
        return 0;
}
```

```
sizeof(his.name) = 13 bytes
sizeof(his.age) = 4 bytes
sizeof(his.phone) = 13 bytes
sizeof(his) = 36 bytes

이름 : benny
나이 : 23
전화번호 : 010-333-4444
his.name=benny, his.age=23, his.phone=010-333-4444
```



[그림 8.1] [예제 8.1] 구조체 멤버 변수의 기억장소 할당 형태

구조체 배열

구조체 배열의 선언과 초기화

```
struct neighbor {
        char name[13];
        int age;
        char phone[13];
};
typedef struct neighbor neighbor;
neighbor our[3] = {
        {“benny”, 23, “010-333-4444”}, //구조체 our[0]
        {“daniel”, 20, “010-555-6666”}, //구조체 our[1]
        {“joon”, 10, “010-777-8888”} //구조체 our[2]
};
```

구조체 배열의 멤버변수 사용 예

```
strcpy(our[0].name, “jhon”);
our[1].age = 27;
gets(our[2].phone);
```

our[0].name="benny" 13bytes	our[0].age=23 4bytes	our[0].phone="010-333-4444" 13bytes
our[1].name="daniel" 13bytes	our[1].age=20 4bytes	our[1].phone="010-555-6666" 13bytes
our[2].name="joon" 13bytes	our[2].age=10 4bytes	our[2].phone="010-777-8888" 13bytes

[그림 8.2] 구조체 배열의 기억장소 할당 형태

구조체 배열의 활용

[예제 8.2] 구조체 배열의 활용

```
#include <stdio.h>
#define SIZE 3

struct neighbor {
	char name[13];
	int age;
	char phone[13];
};
typedef struct neighbor neighbor;

int main(int argc, char *argv[])
{
	neighbor our[SIZE];

	for (int i = 0; i < sizeof(our) / sizeof(neighbor); i++) {
		printf("이름 : ");
		gets(our[i].name);

		printf("나이 : ");
		scanf("%d", &our[i].age);

		while (getchar() != '₩n'); //입력 버퍼에 저장되어 있는 값 삭제

		printf("전화번호 : ");
		gets(our[i].phone);
	}
	puts("");
	for (int i = 0; i < sizeof(our) / sizeof(neighbor); i++) {
		printf("our[%d] = %s, %d, %s₩n", i, our[i].name, our[i].age, our[i].phone);
	}
	return 0;
}
```

```
이름 : benny
나이 : 23
전화번호 : 010-333-4444
이름 : daniel
나이 : 20
전화번호 : 010-555-6666
이름 : joon
나이 : 10
전화번호 : 010-777-8888

our[0] = benny, 23, 010-333-4444
our[1] = daniel, 20, 010-555-6666
our[2] = joon, 10, 010-777-8888
```

구조체 포인터

구조체 포인터의 선언 및 초기화

```
struct neighbor {
        char name[13];
        int age;
        char phone[13];
};
typedef struct neighbor neighbor;
neighbor her, our[3], *h, *o;
h=&her; //포인터 변수에 일반 변수의 주소를 대입
o=our;
```

구조체 포인터의 멤버 참조 (일반 변수)

1) 구조체포인터변수명->멤버변수
2) (*구조체포인터변수명).멤버변수
//'*' 는 '.' 보다 우선순위가 낮기 때문에 괄호 필요

구조체 포인터의 멤버 참조 (배열 변수)

1) (구조체포인터변수명+배열인덱스)->멤버변수
2) 구조체포인터변수명[배열인덱스].멤버변수
3) (*(구조체포인터변수명+배열인덱스)).멤버변수
//'*' 는 '.' 보다 우선순위가 낮기 때문에 괄호 필요

구조체 포인터의 사용 예

```
h->name;        //(*h).name;
h->age;         //(*h).age;
(o+2)->name;    //o[2].name; (*(o+2)).name;
(o+2)->age;     //o[2].age; //(*(o+2)).age;
```

구조체 포인터의 활용

[예제 8.3] 구조체 포인터의 활용

```
#include <stdio.h>
#define SIZE 3

struct neighbor {
        char name[13];
        int age;
        char phone[13];
};
typedef struct neighbor neighbor;

int main(int argc, char *argv[])
{
        neighbor her = { "raina", 25, "010-111-2222" };
        neighbor our[SIZE] = {
                {"benny", 23, "010-111-2222"},
                {"daniel", 20, "010-333-4444"},
                {"joon", 10, "010-555-6666"}
        };
        neighbor *h, *o;

        h = &her;
        o = our;

        printf("h->name=%s, h->age=%d₩n", h->name, h->age);
        printf("(*h).name=%s, (*h).age=%d₩n₩n", (*h).name, (*h).age);

        for(int i=0; i<sizeof(our)/sizeof(neighbor); i++) {
                printf("(o+%d)->name=%s, (o+%d)->age=%d₩n",
                        i, (o+i)->name, i, (o+i)->age);
                printf("o[%d].name=%s, o[%d].age=%d₩n",
                        i, o[i].name, i, o[i].age);
                printf("(*(o+%d)).name=%s, (*(o+%d)).age=%d₩n",
                        i, (*(o+i)).name, i, (*(o+i)).age);
                puts("");
        }
        return 0;
}
```

```
h->name=raina, h->age=25
(*h).name=raina, (*h).age=25

(o+0)->name=benny, (o+0)->age=23
o[0].name=benny, o[0].age=23
(*(o+0)).name=benny, (*(o+0)).age=23

(o+1)->name=daniel, (o+1)->age=20
o[1].name=daniel, o[1].age=20
(*(o+1)).name=daniel, (*(o+1)).age=20

(o+2)->name=joon, (o+2)->age=10
o[2].name=joon, o[2].age=10
(*(o+2)).name=joon, (*(o+2)).age=10
```

중첩된 구조체

중첩된 구조체 예

```
struct birth {
        int year;
        int month;
        int day;
};
typedef struct birth birth;

struct neighbor {
        char name[13];
        int age;
        char phone[13];
        birth birthday;
};
typedef struct neighbor neighbor;
```

중첩된 구조체의 참조

```
neighbor her, our[3];
neighbor *h, *o;
h = &her;
o = our;

her.birthday.year = 2001;
h->birthday.year = 2001;
(o+1)->birthday.year = 2001;
```

tm 구조체

<time.h>의 tm 구조체

```
struct tm {
    int tm_sec;         //초, 0-59
    int tm_min;         //분, 0-59
    int tm_hour;        //시, 0-23
    int tm_mday;        //그 달의 날 수, 1-31
    int tm_mon;         //1월부터 월 수, 0-11
    int tm_year;        //1900년부터 연 수
    int tm_wday;        //일요일부터 요일 수, 0-6
    int tm_yday;        //1월 1일부터 날 수, 0-365
    int tm_isdst;       //절약 시간 지정(썸머타임제 사용 여부)
};
```

time_t 데이터 타입

1970년 1월 1일 자정을 기준으로 현재 시스템에 저장된 시간까지 경과된 시간을 초 단위로 나타낸 값. 64bits 사용하여 값을 저장.

사용법

```
time_t now = time(NULL); //현재 시간을 초 단위로 저장
struct tm *today;
today = localtime(&now); //today의 각 멤버에 현재 시간을 저장

//현재 시간에 해당되는 연, 월, 일
int year = today->tm_year + 1900; //연은 1900을 더함
int month = today->tm_mon + 1; //월은 1을 더함
int day = today->tm_mday;
```

중첩된 구조체의 활용

[예제 8.4] 중첩된 구조체의 활용

```
#include <stdio.h>
#include <time.h>
#define SIZE 3

struct birth {
    int year;
    int month;
    int day;
};
typedef struct birth birth;

struct neighbor {
    char name[13];
    int age;
    char phone[13];
    birth birthday;
};
typedef struct neighbor neighbor;

int main(int argc, char *argv[])
{
    int yyyymmdd;
    neighbor our[SIZE] = {
        {"benny", 0, "010-111-2222"},
        {"daniel", 0, "010-333-4444"},
        {"joon", 0, "010-555-6666"}
    };
    neighbor *o = our;
    time_t now = time(NULL);
    struct tm *today = localtime(&now);

    for (int i = 0; i < sizeof(our) / sizeof(neighbor); i++) {
        printf("%s의 생년월일을 입력해주세요(예: 20010722) : ",
(o+i)->name);
        scanf("%d", &yyyymmdd);

        (o+i)->birthday.year = yyyymmdd / 10000;
        (o+i)->birthday.month = yyyymmdd % 10000 / 100;
        (o+i)->birthday.day = yyyymmdd % 100;
        (o+i)->age = ((today->tm_year)+1900) - (o+i)->birthday.year;
    }
    puts("");
    for (int i = 0; i < sizeof(our) / sizeof(neighbor); i++) {
        printf("%s, %d년 %d월 %d일생, %d세, %s\n",
            (o+i)->name, (o+i)->birthday.year, (o+i)->birthday.month,
            (o+i)->birthday.day, (o+i)->age, (o+i)->phone);
    }

    printf("\n오늘의 날짜 : %d년 %d월 %d일 \n\n",
        (today->tm_year)+1900, today->tm_mon+1, today->tm_mday);

    for (int i = 0; i < sizeof(our) / sizeof(neighbor); i++) {
        if ( (o+i)->birthday.month == (today->tm_mon+1) &&
            (o+i)->birthday.day == today->tm_mday ) {
                printf("%s님의 %d번째 생일을 축하합니다^^\n",
                    (o+i)->name, (o+i)->age);
        }
    }
    return 0;
}
```

```
benny의 생년월일을 입력해주세요(예: 20010722) : 19980416
daniel의 생년월일을 입력해주세요(예: 20010722) : 20011010
joon의 생년월일을 입력해주세요(예: 20010722) : 20100806

benny, 1998년 4월 16일생, 22세, 010-111-2222
daniel, 2001년 10월 10일생, 19세, 010-333-4444
joon, 2010년 8월 6일생, 10세, 010-555-6666

오늘의 날짜 : 2020년 8월 6일

joon님의 10번째 생일을 축하합니다^^
```

자기 참조 구조체

자기 참조 구조체의 선언

```
struct neighbor {
        char name[13];
        int age;
        struct neighbor *next; //자기 참조
};
typedef struct neighbor neighbor;
```

자기 참조 구조체의 사용

```
neighbor her, his;
her->next = his;
```

[예제 8.5] 자기 참조 구조체의 활용

```
#include <stdio.h>

struct neighbor {
        char name[13];
        int age;
        struct neighbor *next; //자기 참조
};
typedef struct neighbor neighbor;

int main(int argc, char *argv[])
{
        neighbor *p, a={"benny", 23}, b={"daniel", 20}, c={"joon", 10};

        p = &a;
        a.next = &b;
        b.next = &c;
        c.next = '₩0';

        while (p != NULL) {
                printf("%s, %d₩n", p->name, p->age);
                p = p->next;
        }
}
```

```
benny, 23
daniel, 20
joon, 10
```

함수에서 구조체 전달

함수에서의 구조체 멤버 변수 전달

```
#define SIZE 3
struct neighbor {
        char name[13];
        int age;
        int yymmdd[SIZE]
};
typedef struct neighbor neighbor;
neighbor her, our[SIZE];
```

함수의 원형 예	함수의 호출 예
neighbor func1(void);	her = func1(); our[1] = func1();
void func2(neighbor data);	func2(her); func2(our[1]);

void func3(neighbor *data);	func3(&her);
	func3(&our[1]);
	func3(our);

void func4(int *data, int size);	func4(her.yymmdd, SIZE); func4(our[1].yymmdd, SIZE);
void func5(int *data);	func5(&her.age); func5(&our[1].age);
void func6(char *data);	func6(her.name); func6(our[1].name);

함수에서 구조체의 전달과 반환

[예제 8.6] 함수에서 구조체의 전달과 반환

```
#include <stdio.h>
#include <string.h>
#define SIZE 3

struct neighbor {
      char name[13];
      int age;
      char phone[13];
};
typedef struct neighbor neighbor;

neighbor find(neighbor *data, int size, char *n) //사용자 정의 함수
{
      neighbor tmp = {"NULL", 0, "NULL"};
      for (int i = 0; i < size; i++) {
          if (!strcmp((data + i)->name, n))
                  return (data[i]);
      }
      return tmp;
}
```

```
이름을 입력하시면 해당 정보를 찾아드립니다.
이름을 입력하세요 : daniel
 => daniel, 20세, 010-333-4444입니다.
```

```
이름을 입력하시면 해당 정보를 찾아드립니다.
이름을 입력하세요 : raina
 => 정보가 없습니다.
```

```
int main(int argc, char *argv[])
{
        char name[13]; //구조체의 멤버변수 name과 변수 name은 별개임.
        neighbor who;
        neighbor our[SIZE] = {
                {"benny", 23, "010-111-2222"},
                {"daniel", 20, "010-333-4444"},
                {"joon", 10, "010-555-6666"}
        };

        printf("이름을 입력하시면 해당 정보를 찾아드립니다.\n");
        printf("이름을 입력하세요 : ");
        gets(name);

        who = find(our, SIZE, name);
        if (!strcmp(who.name, "NULL"))
                printf(" => 정보가 없습니다.\n");
        else
                printf(" => %s, %d세, %s입니다.\n", who.name, who.age,
who.phone);
        return 0;
}
```

비트 필드 구조체 형식

비트 필드 구조체 형식

```
struct 구조체명칭 {
    데이터형 비트변수1 : 비트길이;
    데이터형 비트변수2 : 비트길이;
      …                          ;
    데이터형 비트변수n : 비트길이;
};
```

비트 필드 구조체 예

```
struct bit_field {
      unsigned a : 1; //1bit, 가질 수 있는 값은 0 또는 1
      unsigned b : 1;
      unsigned c : 1;
      unsigned d : 2; //2bits, 가질 수 있는 값은 0, 1, 2, 3
      unsigned e : 1;
      unsigned f : 1;
      unsigned g : 1;
} flag = {1, 0, 0, 3, 0, 1, 1};
```

flag(1 byte)의 저장 값과 메모리 형태

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	0	1	1	0	1	1
g	f	e	d		c	b	a

공용체 정의 및 메모리 구조

□ 공용체 정의 및 메모리 구조

1) 공용체 정의 예

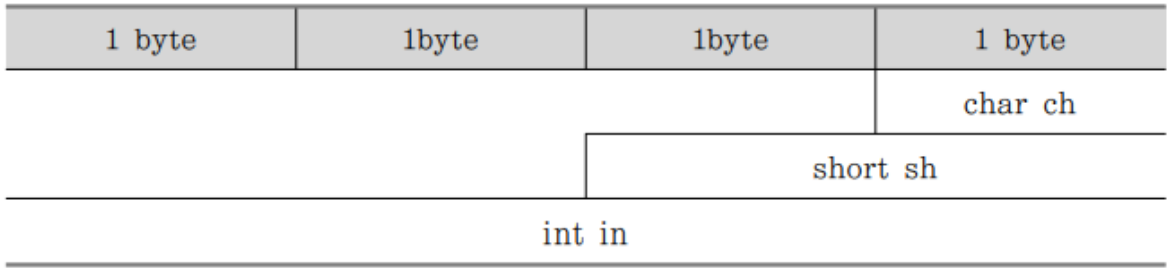
```
union u_tag {
      char ch;
      short sh;
      int in;
} info_u
```

2) 구조체 정의 예

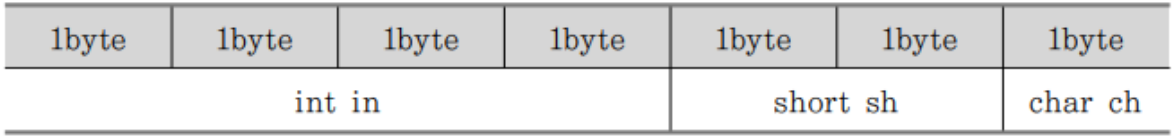
```
struct s_tag {
      char ch;
      short sh;
      int in;
} info_s;
```

3) 공용체 및 구조체 변수 info_u, info_s의 메모리 구조

• 공용체 info_u의 메모리 구조



• 구조체 info_s의 메모리 구조



공용체의 활용

[예제 8.7] 공용체의 활용

```
#include <stdio.h>

int encode_decode(int i)
{
        char ch;
        union encodeco {
                int num;
                char ch[4];
        } ende;
        ende.num = i;

        //바이트를 서로 교환해서 암호화하고, 암호화된 정수를 복호화 한다.
        ch = ende.ch[0];
        ende.ch[0] = ende.ch[1];
        ende.ch[1] = ch;

        ch = ende.ch[2];
        ende.ch[2] = ende.ch[3];
        ende.ch[3] = ch;

        return ende.num; //암호화된 정수를 반환한다.
}
```

```
암호화하실 숫자를 입력 하세요(종료 0) : 25
25 encoded is 6400
6400 decoded is 25
암호화하실 숫자를 입력 하세요(종료 0) : 7
7 encoded is 1792
1792 decoded is 7
암호화하실 숫자를 입력 하세요(종료 0) : 0
```

```
int main(int argc, char *argv[])
{
    int en, de, num;

    while (1) {
        printf("암호화하실 숫자를 입력 하세요(종료 0) : ");
        scanf("%d", &num);

        if (num == 0) break;

        en = encode_decode(num); //암호화 한다.
        printf("%d encoded is %d₩n", num, en);

        de = encode_decode(en); //복호화 한다.
        printf("%d decoded is %d₩n", en, de);
    }
    return 0;
}
```

구조체와 공용체의 응용 (1)

[예제 8.8] 구조체를 포함한 공용체의 활용

```
#include <stdio.h>

struct binary {
        //char는 1byte(8bits)
        unsigned b0: 1;  //LSB(가장 뒤)의 첫 번째 비트를 표현
        unsigned b1: 1;  //LSB의 두 번째 비트를 표현
        unsigned b2: 1;
        unsigned b3: 1;
        unsigned b4: 1;
        unsigned b5: 1;
        unsigned b6: 1;
        unsigned b7: 1; //MSB(가장 앞)의 첫 번째 비트를 표현
};
union byteint {
        char ch; //bits구조체를 char 한 글자로 표현
        struct binary bits;
} byte;
```

```
-128~127 사이의 정수를 이진수로 변환합니다.
정수를 입력하세요 : 321
다시 입력하세요 : -369
다시 입력하세요 : 16
이진수 = 0 0 0 1 0 0 0 0
```

```
int main(int argc, char *argv[])
{
        int num;

        printf("-128~127 사이의 정수를 이진수로 변환합니다.\n");
        printf("정수를 입력하세요 : ");

        while (1) {
                scanf("%d", &num);
                if (-128 <= num && num <= 127) break;
                printf("다시 입력하세요 : ");
        }
        byte.ch = num;

        printf("이진수 = ");
        printf("%d ", byte.bits.b7? 1:0);
        printf("%d ", byte.bits.b6? 1:0);
        printf("%d ", byte.bits.b5? 1:0);
        printf("%d ", byte.bits.b4? 1:0);
        printf("%d ", byte.bits.b3? 1:0);
        printf("%d ", byte.bits.b2? 1:0);
        printf("%d ", byte.bits.b1? 1:0);
        printf("%d ", byte.bits.b0? 1:0);
        return 0;
}
```

구조체와 공용체의 응용 (2)

[예제 8.9] 구조체 이용한 진법 변환

```
#include <stdio.h>
#include <string.h> //strcpy(), strcat(), strlen(), strcmp()

struct binary2hexa{
      char hexa[2];
      char bin[5];
};
typedef struct binary2hexa binary2hexa;
```

```
4비트씩의 2진수 입력 : 0011
  -> 16진수 변환 값은 = 0x3

4비트씩의 2진수 입력 : 00111111
  -> 16진수 변환 값은 = 0x3F

4비트씩의 2진수 입력 : 111100001111
  -> 16진수 변환 값은 = 0xF0F

4비트씩의 2진수 입력 : 2
  -> 2진수 입력이 올바르지 않아 종료합니다.
```

```
void makehexa(char *in, char *rst, binary2hexa *b2h, int size)
{
      int i=0, j=0;
      char temp[5]="";

      while( *(in+i) ) {
            if( *(in+i)<'0' || *(in+i)>'1' ) {
                  *(rst+0)='x';
                  return; //2진수가 아니면 리턴
            }
            i++;
      }
      if( (i%4)!=0 ) {
            *(rst+0)='x';
            return; //4비트씩이 아니면 리턴
      }
      i=0;
      while( *(in+i) ) {
            for(j=0; j<4; j++) {
                  *(temp+j) = *(in+i);
                  i++;
            }
            temp[j] = '\0'; //4개 씩 담고

            for(j=0; j<size; j++) {
                  if( strcmp((b2h+j)->bin, temp)==0 )
                        strcat(rst, (b2h+j)->hexa);
            } //배열에 있는지 찾아서 rst에 저장
      }
}
```

```
int main(int argc, char *argv[])
{
      char input[50];
      char result[10];
      binary2hexa b2h[16] = {
          "0", "0000", "1", "0001", "2", "0010", "3", "0011",
          "4", "0100", "5", "0101", "6", "0110", "7", "0111",
          "8", "1000", "9", "1001", "A", "1010", "B", "1011",
          "C", "1100", "D", "1101", "E", "1110", "F", "1111"};

      do {
          result[0]='₩0';
          printf("4비트씩의 2진수 입력 : ");
          gets(input);

          makehexa(input, result, b2h, sizeof(b2h)/sizeof(b2h[0]))

          if(result[0]=='x') break;

          printf("  -> 16진수 변환 값은 = 0x%s₩n₩n", result);

      } while(1);

      printf("  -> 2진수 입력이 올바르지 않아 종료합니다.₩n₩n");
      return 0;
}
```

개념 확인학습 & 적용 확인학습 & 응용 프로그래밍

- 다음 파일에 있는 문제들의 해답을 스스로 작성 해 보신 후 개념 & 적용 확인 학습 영상을 학습 하시기 바랍니다.
  - c_08장_구조체공용체_ex.pdf
- 퀴즈와 과제가 출제되었습니다.
  - 영상을 학습하신 후 과제와 퀴즈를 수행 하시기 바랍니다.

Q & A

- "포인터"에 대한 학습이 모두 끝났습니다.

- 모든 내용을 이해 하셨나요?
- 아직 이해가 안되는 내용이 있다면 다시 한번 복습하시기 바랍니다.
- 질문은 한림 SmartLEAD 쪽지 또는 e-mail 또는 전화상담을 이용하시기 바랍니다.

- 퀴즈와 과제가 출제되었습니다. 마감시간에 늦지 않도록 주의해 주세요.

- 다음 시간에는 "파일입출력"에 대해 알아보겠습니다.

- 수고하셨습니다.^^