프로그래밍언어

9. 구조체 (Structure), 구조체 배열, 바이트 저장순서, 비트 단위 구조체

교수 김 영 탁

영남대학교 기계IT대학 정보통신공학과

(Tel : +82-53-810-2497; Fax : +82-53-810-4742

http://antl.yu.ac.kr/; E-mail : ytkim@yu.ac.kr)

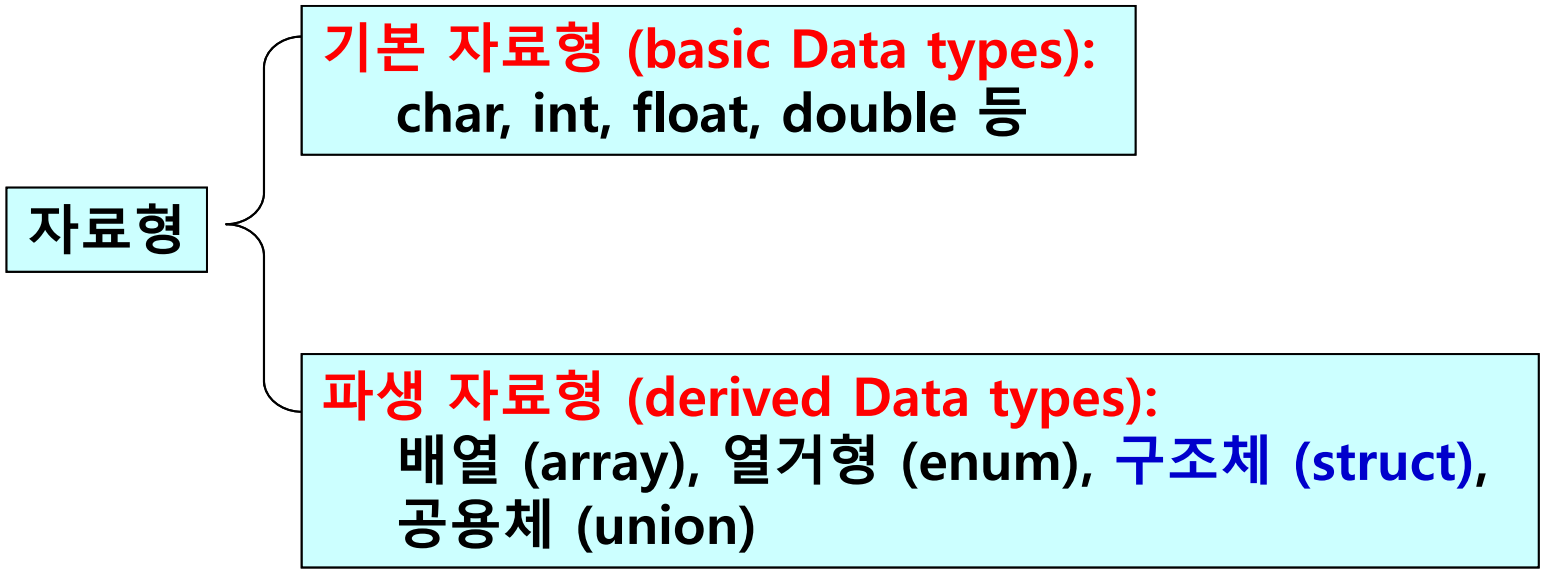
Outline

- 구조체란 무엇인가 ?
- 구조체의 선언
- 자료형 지정 (typedef)
- 구조체 초기화, 응용
- 구조체의 배열
- 구조체와 포인터
- 구조체 포인터의 함수 인수 전달
- 바이트 저장 순서 (Byte ordering)
- Bit-field 구조체
- 자기참조 구조체

구조체 (Structure) 란?

자료형의 분류

◆ Data Types



구조체 (struct)의 필요성

◆ 학생에 대한 다양한 데이터를 모와 묶음으로 사용하려면?

학번: 21900001 (integer)
성명: 김영웅 (string)
학점: 96.8 (double)

```
int Student_id;
char name[10];
double gpa; // grade point average
```

int Student_id;
char name[10];
double gpa;
와 같이 개별 변수로
나타낼 수 있지만
묶음으로 만들어 사용
할 수가 없나?

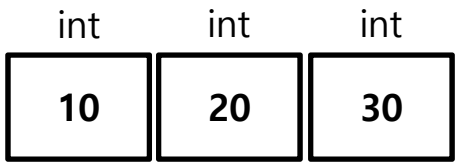
구조체 (struct Student)

st_id	name	gpa

구조체와 배열

◆ 구조체 vs 배열

```
int Data[50];
```



(a) 배열 : 같은 자료형 데이터의 집합

```
struct  Student
{
    int st_id;
    char  name[50];
    double  gpa;
    . . . .
};
```

(b) 구조체 : 다른 자료형 데이터의 집합

구조체 선언

◆ 구조체 선언 형식

```
struct 태그 {
      자료형    멤버1;
      자료형    멤버2;
      …
};
```

(주의사항)
- 구조체 선언은 새로운 자료형 (Data type)을 선언하는 것이며, 변수(variable)을 선언하는 것이 아님
- 구조체 변수를 선언하기 위해서는 구조체를 자료형으로 변수를 별도로 선언하여야 함

```
struct Student  // 구조체 선언
{
   int st_id;  // 구조체 Student의 속성 (attribute)
   char name[10];
   double gpa;  // grade point average
   Date birth_date; // 생년월일
   Tel_Number tel_no; // 다른 구조체의 변수를 속성으로 포함
};

struct Student  st1, st2;  // 구조체 변수 선언
struct Student  st_array[50]; // 구조체 배열 선언
```

구조체 선언의 예

```
// x값과 y값으로 이루어지는 화면의 좌표
struct Point {
    int x;          // x 좌표
    int y;          // y 좌표
};
```

```
struct Complex {// 복소수
    double real;    // 실수부
    double imag;    // 허수부
};
```

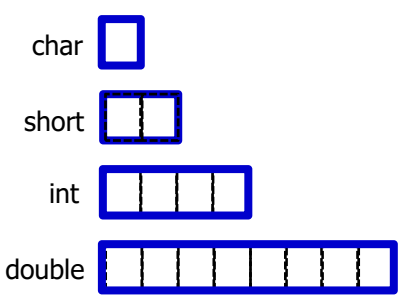
```
struct Rect {// 사각형 도형
    int x;
    int y;
    int width;
    int length;
};
```

```
struct Date {// 날짜
    int year;
    int month;
    int day;
};
```

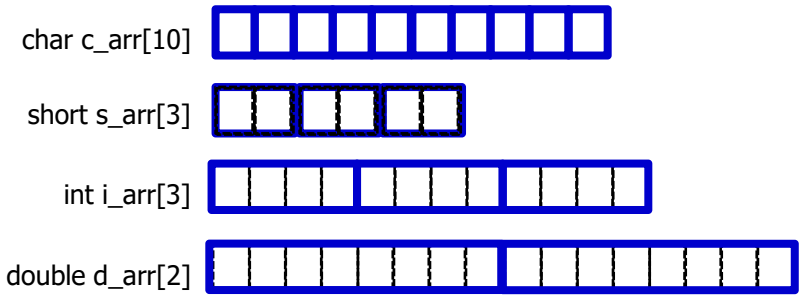
```
struct Tel_Num { // 전화번호
    unsigned short nation_code;
    unsigned short region_code;
    unsigned short switch_no;
    unsigned short line_no;
};
```

```
// 학생
struct Student {
    char name[20]; // 이름
    int st_id;          // 학번
    double GPA;     // 성적 평점
    struct Date dob;// date_of_birth
     struct Tel_Num tel_no; // 연락처
};
```

배열과 구조체의 비교

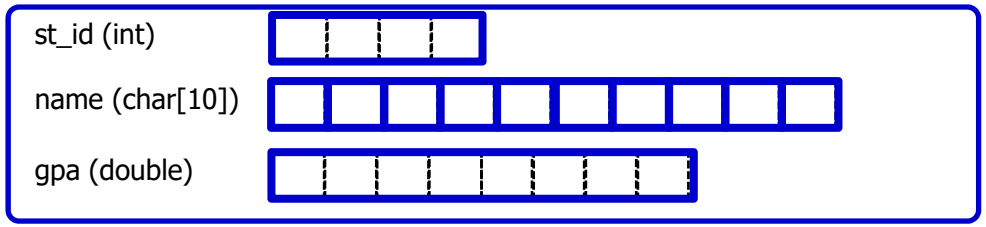


(a) 기본 자료형

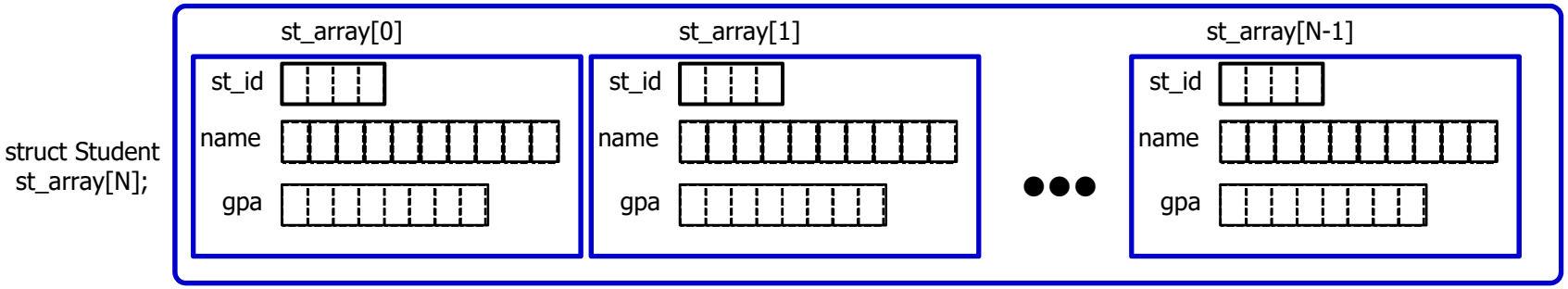


(b) 기본자료형의 배열

```
struct Student
{
    int st_id;
    char name[10];
    double gpa;
};
```



(c) 구조체



(d) 구조체 배열

구조체 변수 선언

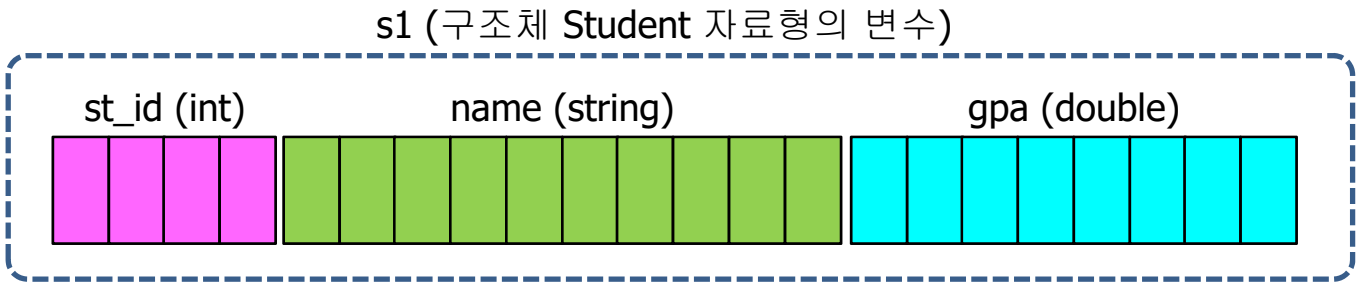
◆ 구조체 정의 (새로운 자료 타입 선언)과
구조체 변수 선언 (자료 타입을 사용한 변수 생성)은 별도

```
struct Student
{
   int st_id;
   char name[10];
   double gpa;
};
```

구조체 정의

```
int main(void)
{
    struct Student s1;
    . . . . .
}
```

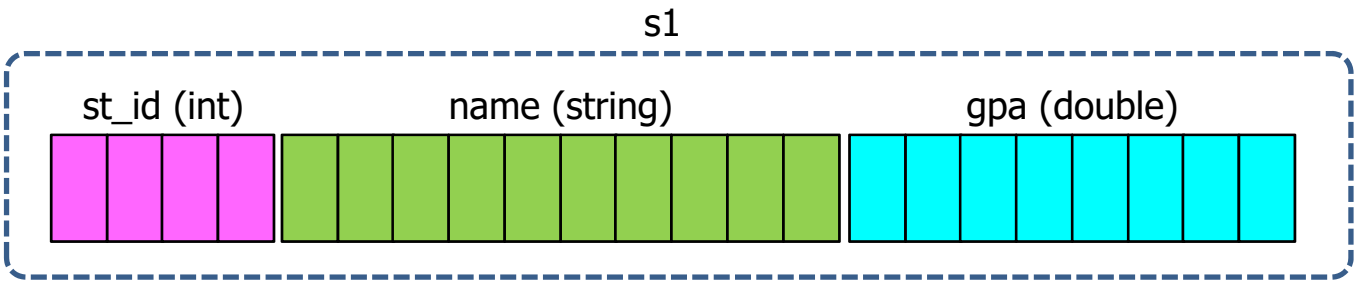
구조체 변수 선언



구조체 변수의 초기화

◆ 중괄호를 이용하여 초기값을 구조체 항목 순서대로 나열

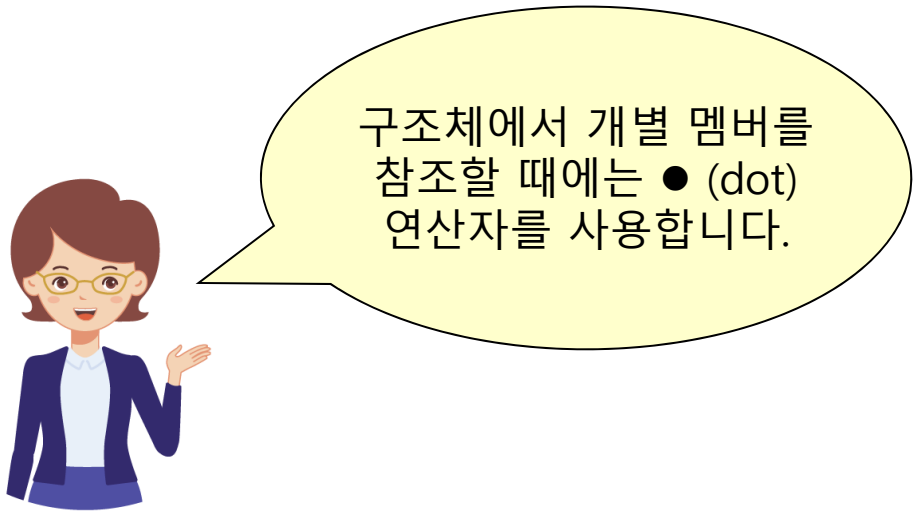
```
struct Student {
        int st_id;
        char name[10];
        double gpa;
};
struct Student s1 = { 24, "Kim", 4.3 };
```



구조체 멤버 참조

◆ 구조체 멤버를 참조하려면 다음과 같이 '.' 연산자를 사용한다.

```
s1.st_id = 21901000;        // 정수 멤버
strcpy(s1.name, "Kim");     // 문자열 멤버
s1.gpa = 4.3;               // 실수 멤버
```



예제 #1

```
...
struct Student {
    int st_id;
    char name[10];
    double gpa;
};

int main(void)
{
    struct Student s;

    s.st_id = 20070001;
    strcpy(s.name,"홍길동");
    s.gpa = 4.3;

    printf("학번: %d\n", s.st_id);
    printf("이름: %s\n", s.name);
    printf("학점: %lf\n", s.gpa);
    return 0;
}
```

구조체 선언

구조체 변수 선언

구조체 멤버 참조

학번: 20070001
이름: 홍길동
학점: 4.300000

예제 #2

```
struct Student {
    int st_id;
    char name[10];
    double gpa;
};

int main(void)
{
    struct Student s;

    printf("학번을 입력하시오: ");
    scanf("%d", &s.st_id);

    printf("이름을 입력하시오: ");
    scanf("%s", s.name);

    printf("학점을 입력하시오(실수): ");
    scanf("%lf", &s.gpa);

    printf("학번: %d\n", s.st_id);
    printf("이름: %s\n", s.name);
    printf("학점: %lf\n", s.gpa);
    return 0;
}
```

구조체 선언

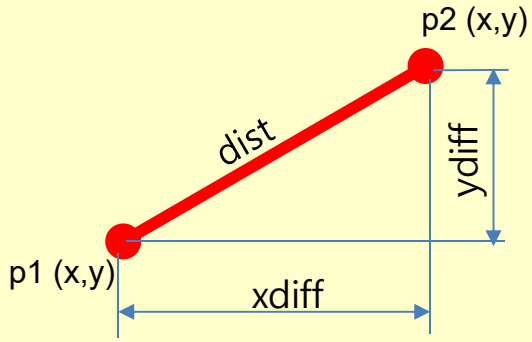
구조체 변수 선언

구조체 멤버의 주소 전달

학번을 입력하시오: 20070001
이름을 입력하시오: 홍길동
학점을 입력하시오(실수): 4.3
학번: 20070001
이름: 홍길동
학점: 4.300000

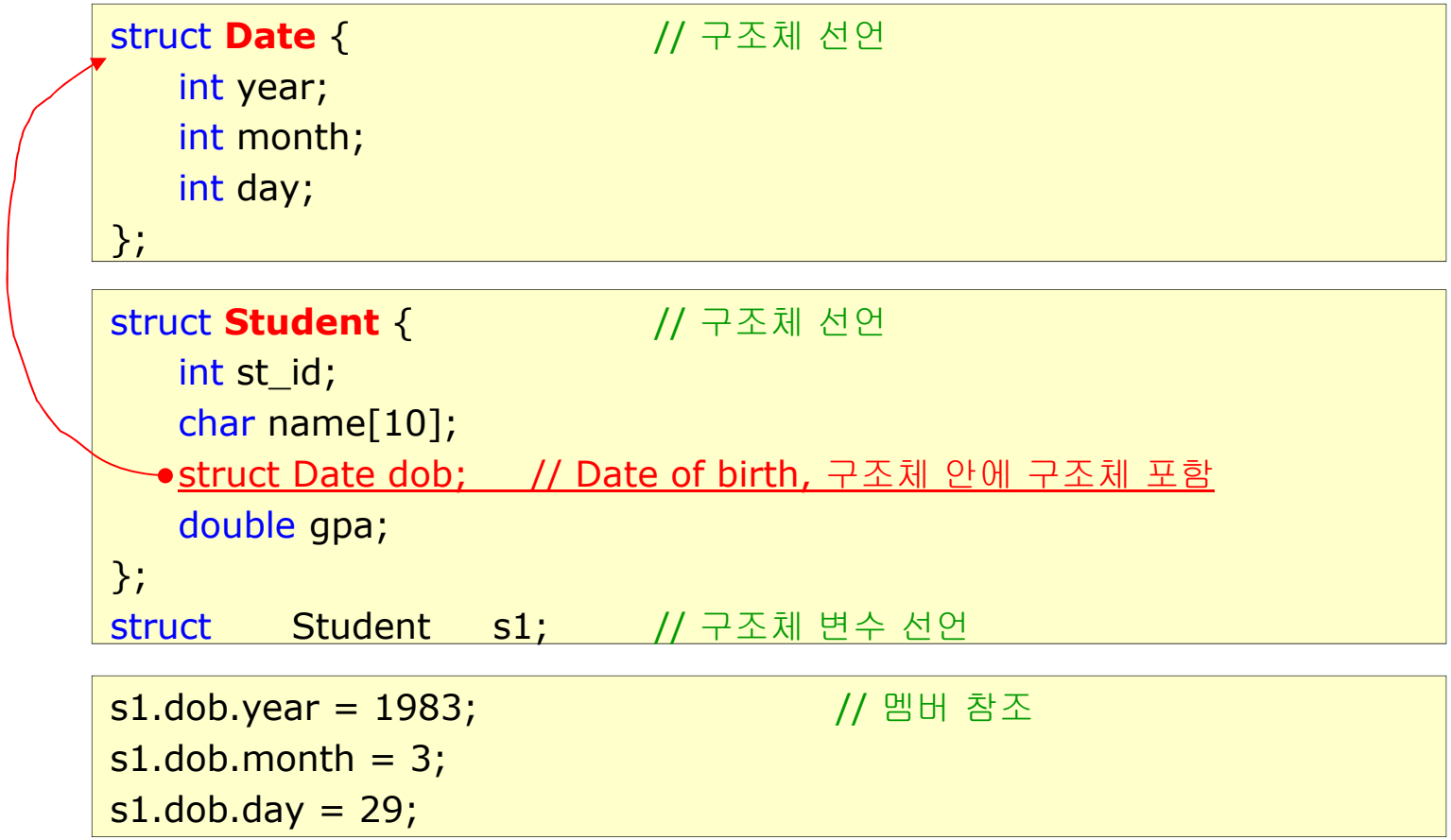
예제 #3

```
#include <math.h>
struct Point {
    int x;
    int y;
};

int main(void)
{
    struct Point p1, p2;
    int xdiff, ydiff;
    double dist;

    printf("점의 좌표를 입력하시오(x  y): ");
    scanf("%d %d", &p1.x, &p1.y);

    printf("점의 좌표를 입력하시오(x  y): ");
    scanf("%d %d", &p2.x, &p2.y);

    xdiff = p1.x - p2.x;
    ydiff = p1.y - p2.y;

    dist = sqrt(xdiff * xdiff + ydiff * ydiff);

    printf("두 점사이의 거리는 %lf입니다.\n", dist);
    return 0;
}
```




```
점의 좌표를 입력하시오(x y): 10 10
점의 좌표를 입력하시오(x y): 20 20
두 점사이의 거리는 14.142136입니다.
```

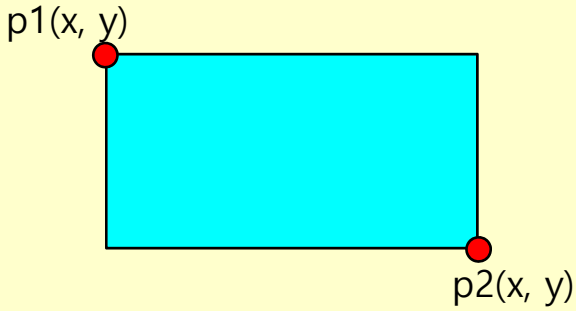
구조체 변수를 멤버로 가지는 구조체



```
struct Date {                       // 구조체 선언
    int year;
    int month;
    int day;
};

struct Student {                    // 구조체 선언
    int st_id;
    char name[10];
    struct Date dob;    // Date of birth, 구조체 안에 구조체 포함
    double gpa;
};
struct    Student    s1;            // 구조체 변수 선언

s1.dob.year = 1983;                 // 멤버 참조
s1.dob.month = 3;
s1.dob.day = 29;
```

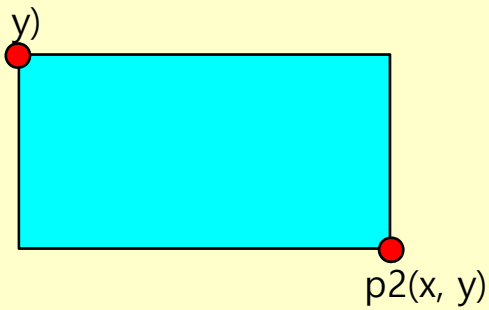
예제

```
#include <stdio.h>

struct Point {
    int x;
    int y;
};

struct Rect {
    struct Point p1;
    struct Point p2;
};

int main(void)
{
    struct Rect r;
    int width, length, area, peri;
```



예제

```
    printf("직사각형 왼쪽 상단의 좌표를 입력하시오: ");
    scanf("%d %d", &r.p1.x, &r.p1.y);

    printf("직사각형 오른쪽 하단의 좌표를 입력하시오: ");
    scanf("%d %d", &r.p2.x, &r.p2.y);

    width = r.p2.x - r.p1.x;
    length = r.p2.y - r.p1.y;

    area = width * length;
    peri = 2 * width + 2 * length;
    printf("직사각형의 면적은 %d이고 둘레는 %d입니다.\n", area, peri);

    return 0;
}
```




```
직사각형 왼쪽 상단의 좌표를 입력하시오: 1 1
직사각형 오른쪽 하단의 좌표를 입력하시오: 6 6
직사각형의 면적은 25이고 둘레는 20입니다.
```

구조체 변수의 대입과 비교

◆ 같은 구조체 변수끼리 직접 대입 (assignment)은 가능하지만 비교 (comparison)은 개별 항목 마다 별도로 실행하여야 함

```
struct Point {
    int x;
    int y;
};

int main(void)
{
    struct Point p1 = {10, 20};
    struct Point p2 = {30, 40};

    p2 = p1;                                  // 대입 가능

    if( p1 == p2 )                            // 비교 --> 컴파일 오류!!
        printf("p1와 p2는 같습니다.")

    if( (p1.x == p2.x) && (p1.y == p2.y) ) // 올바른 비교
        printf("p1와 p2는 같습니다.")
}
```

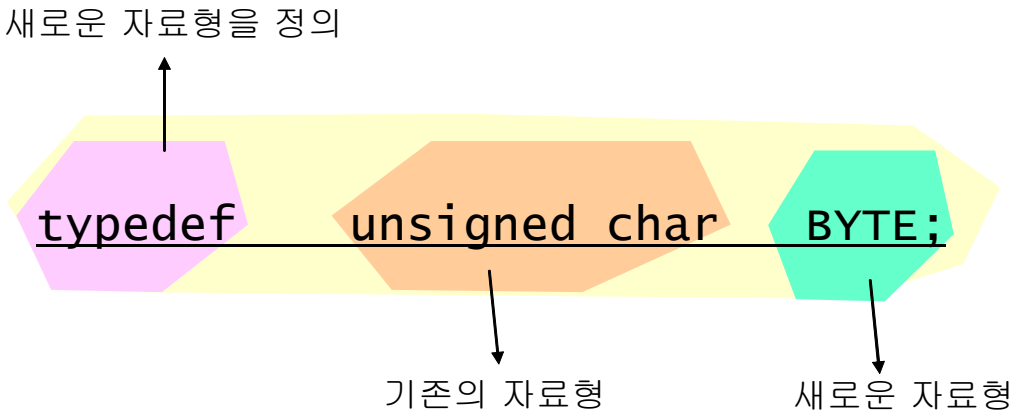
자료형 정의 (typedef)를 사용한 구조체 이름 간소화

typedef

◆ typedef의 의미

- typedef은 새로운 자료형(type)을 정의(define)
- C의 기본 자료형을 확장시키는 역할

```
typedef      old_type      new_type;
```



typedef의 예

```
typedef  int  INT32;
typedef  unsigned int UINT32;
typedef  unsigned short UINT16;
typedef   unsiged char   BYTE; // or OCTET


INT32 i;          // int i; (signed 32-bit)
UINT32 k;         // unsigned int k; (unsigned 32-bit)
UINT16 n;         // unsigned short n;
BYTE index;       // unsigned char index;
```

구조체로 새로운 타입 정의

```c
#include <stdio.h>

typedef struct {
    int x;
    int y;
} POINT;

POINT translate(POINT p, POINT delta);

int main(void)
{
    POINT p = { 2, 3 };
    POINT delta = { 10, 10 };
    POINT result;

    result = translate(p, delta);
    printf("새로운 점의 좌표는(%d, %d)입니다.\n", result.x, result.y);

    return 0;
}
```

예제

```
POINT translate(POINT p, POINT delta)
{
    POINT new_p;

    new_p.x = p.x + delta.x;
    new_p.y = p.y + delta.y;

    return new_p;
}
```

새로운 점의 좌표는 (12, 13)입니다.

typedef 사용의 장점

◆ 이식성 (portability)을 높여준다.

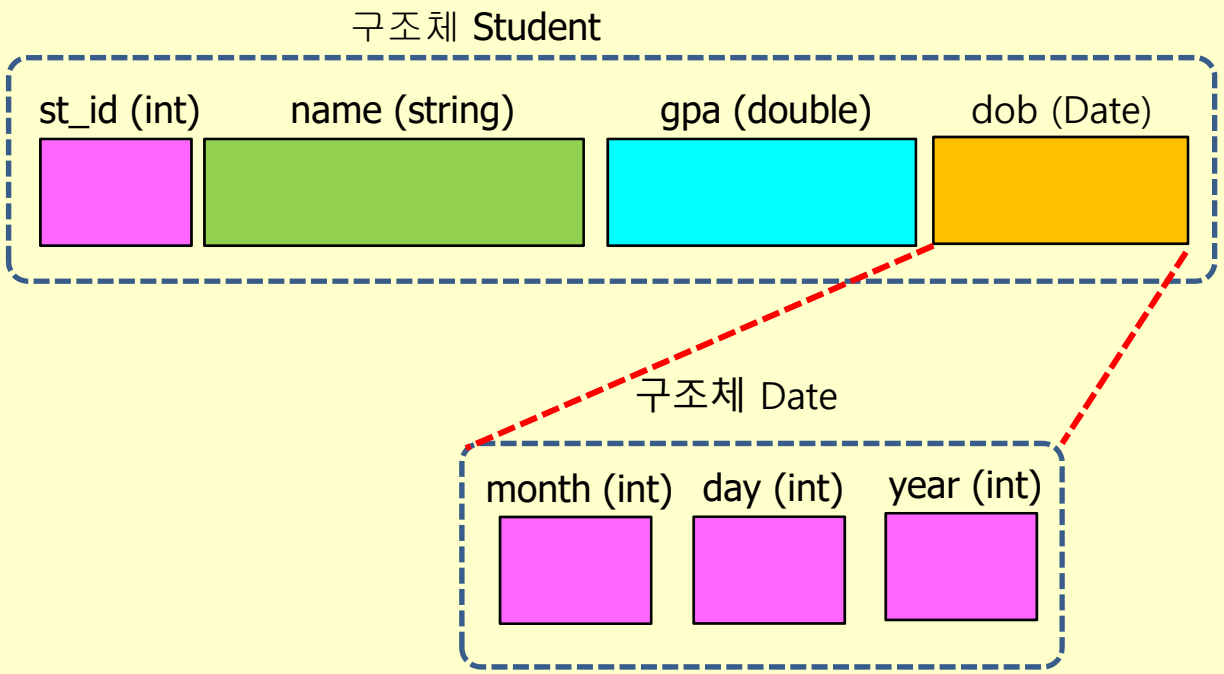
- 소스 코드를 컴퓨터 하드웨어에 독립적으로 만들 수 있다
- (예) int형은 2바이트이기도 하고 4바이트, int형 대신에 typedef을 이용한 INT16나 INT32을 사용하게 되면 확실하게 2바이트 (16비트) 인지 4바이트 (32비트)인지를 지정할 수 있다.

◆ 문서화의 역할도 한다.

- typedef을 사용하게 되면 주석을 붙이는 것과 같은 효과

구조체 내에 다른 구조체 변수를 멤버로 가지는 구조체

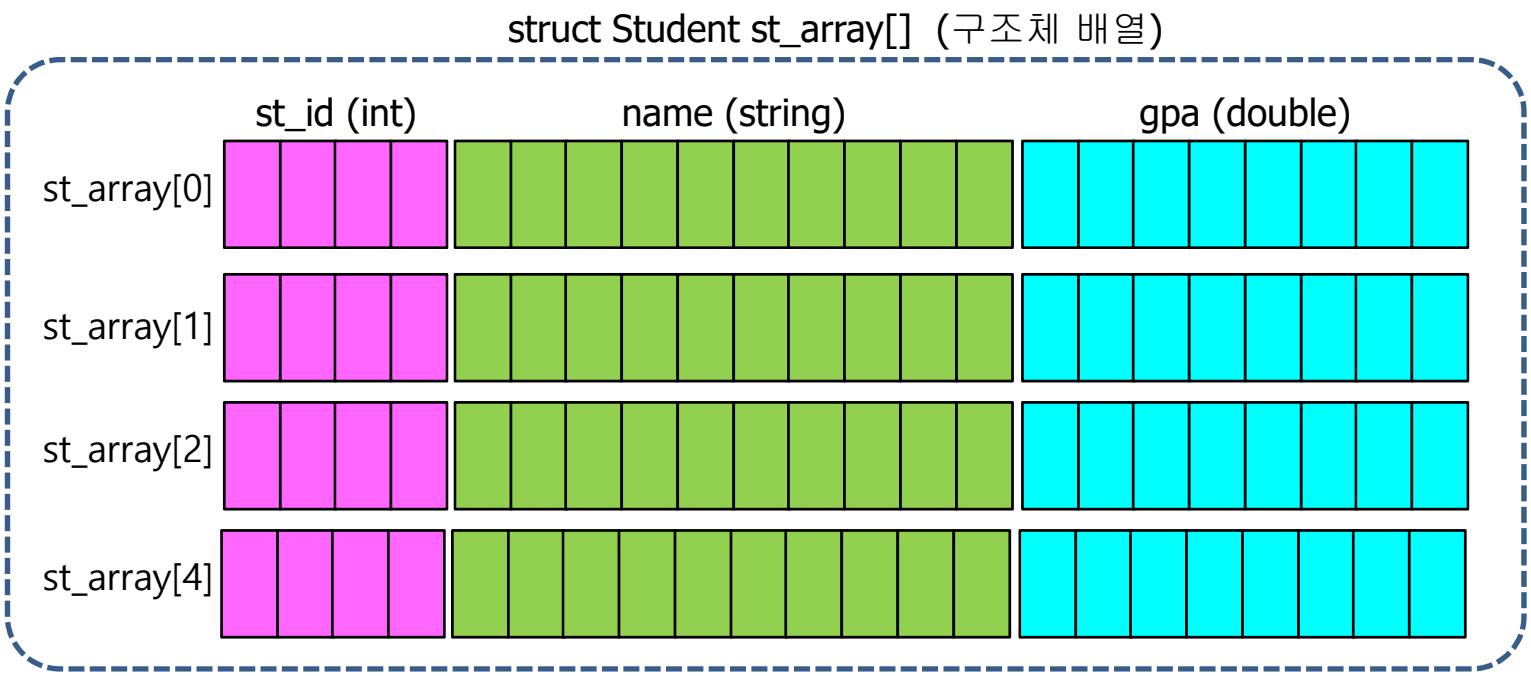
```
typedef struct
{
    int year;
    int month;
    int day;
} Date;

typedef struct
{
    int st_id;
    char name[20];
    double gpa;
    Date dob; // Date of birth
} Student;
```



구조체 배열

구조체 배열 (Array of Struct)

◆ 구조체를 여러 개 모은 것



구조체 배열 (Array of Struct)

◆ 구조체 배열의 선언

```
typedef struct {
    int st_id;   // Student identifier
    char name[20];
    double gpa;  // grade point average
} Student;

int main(void)
{
    Student st_array[100];          // 구조체의 배열 선언

    st_array[2].st_id = 27;
    strcpy(st_array[2].name, "홍길동");
    st_array[2].gpa = 178.0;
}
```

구조체 배열의 초기화

◆ 구조체 배열의 초기화

```
typedef struct {
    int st_id;
    char name[20];
    double gpa;
} Student;

Student st_array[3] = {
    { 1, "Park", 172.8 },
    { 2, "Kim", 179.2 },
    { 3, "Lee", 180.3 }
};
```

구조체 배열 예제

```
#define SIZE 3
typedef struct {
    int st_id;
    char name[20];
    double gpa;
} Student;
int main(void)
{
    Student st_array[SIZE];
    int i;

    for(i = 0; i < SIZE; i++)
    {
        printf("학번을 입력하시오: ");
        scanf("%d", &st_array [i].st_id);
        printf("이름을 입력하시오: ");
        scanf("%s", st_array[i].name);
        printf("학점을 입력하시오(실수): ");
        scanf("%lf", &st_array [i].gpa);
    }
    for(i = 0; i< SIZE; i++)
        printf("학번: %d, 이름: %s,  학점: %lf\n", st_array[i].st_id,
            st_array [i].name, st_array[i].gpa);
    return 0;
}
```

```
학번을 입력하시오: 20070001
이름을 입력하시오: 홍길동
학점을 입력하시오(실수): 4.3
학번을 입력하시오: 20070002
이름을 입력하시오: 김유신
학점을 입력하시오(실수): 3.92
학번을 입력하시오: 20070003
이름을 입력하시오: 이성계
학점을 입력하시오(실수): 2.87
학번: 20070001, 이름: 홍길동, 학점: 4.300000
학번: 20070002, 이름: 김유신, 학점: 3.920000
학번: 20070003, 이름: 이성계, 학점: 2.870000
```

구조체 배열의 응용 예

◆태양계 (Solar System)의 행성 (Planet)

```
typedef struct {
   char name[10];
   double relativeMass;
   double distance; // distance from sun
} Planet;
Planet solarSystem[] =
{
   {"Mercury", 0.0558, 57.9},
   {"Venus", 0.815, 108},
   {"Earth",1.0, 150},
   {"Mars", 0.107, 228},
   {"Jupiter", 318, 778},
   {"Saturn", 95.1, 1430},
   {"Uranus", 14.5, 2870},
   {"Neptune", 17.2, 4500},
   {"Pluto", 0.11, 5900}
};
```

◆ 구조체 배열의 각 원소가 가지는 멤버의 값을 처리하는 예제

```
void printPlanets(Planet solarPlanets[], int num_planet)
{
   for (int i = 0; i<num_planet; i++)
   {
      printf("%2d", i);
      printf("  Name: ");
      printf("%-8s", solarPlanets[i].name);
      printf(" Rel Mass: ");
      printf("%7.3lf", solarPlanets[i].relativeMass);
      printf(" Dist from Sun: ");
      printf("%6.1lf\n", solarPlanets[i].distance);
   }  // end for
}
```

구조체 배열의 정렬

```
void sortPlanetsByRelMass(Planet solarPlanets[], int num_planet)
{
    Planet temp;
    int i, j, m;
    double min_RelMass;
    for (i=0; i<num_planet-1; i++) {
        m = i;
        min_RelMass = solarPlanets[i].relativeMass;
        for (j=i+1; j<num_planet; j++) {
            if (min_RelMass > solarPlanets[j].relativeMass) {
                m = j;
                min_RelMass = solarPlanets[j].relativeMass;
            }
        } // end inner for
        if (m != i) { // if new minimum found, swap
            temp = solarPlanets[i];
            solarPlanets[i] = solarPlanets[m];
            solarPlanets[m] = temp;
        }
    } // end outer for
}
```

```
void sortPlanetsByName(Planet solarPlanets[], int num_planet)
{
    Planet temp;
    int i, j, m;
    char min_Name[10] = {'\0'};
    for (i=0; i<num_planet-1; i++) {
        m = i;
        strcpy(min_Name, solarPlanets[i].name);
        for (j=i+1; j<num_planet; j++) {
            if (strcmp(min_Name, solarPlanets[j].name) > 0) {
                m = j;
                strcpy(min_Name, solarPlanets[j].name);
            }
        } // end inner for
        if (m != i) { // if new minimum found, swap
            temp = solarPlanets[i];
            solarPlanets[i] = solarPlanets[m];
            solarPlanets[m] = temp;
        }
    } // end outer for
}
```

```
int main()
 {
     Planet solarSystem[SOLAR_PLANETS] =
     { {"Mercury", 0.0558, 57.9}, {"Venus", 0.815, 108}, {"Earth",1.0, 150},
       {"Mars", 0.107, 228}, {"Jupiter", 318, 778}, {"Saturn", 95.1, 1430},
       {"Uranus", 14.5, 2870}, {"Neptune", 17.2, 4500}, {"Pluto", 0.11, 5900} };

     printf("\n Initial state\n");
     printPlanets(solarSystem, SOLAR_PLANETS);

     sortPlanetsByRelMass(solarSystem, SOLAR_PLANETS);
     printf("\n After sorting by relative mass:\n");
     printPlanets(solarSystem, SOLAR_PLANETS);

     sortPlanetsByDist(solarSystem, SOLAR_PLANETS);
     printf("\n After sorting by distance from sun:\n");
     printPlanets(solarSystem, SOLAR_PLANETS);

     sortPlanetsByName(solarSystem, SOLAR_PLANETS);
     printf("\n After sorting by name using strcmp and strcpy:\n");
     printPlanets(solarSystem, SOLAR_PLANETS);
     printf("\n\n");
     return 0;
}
```

◆ result of execution

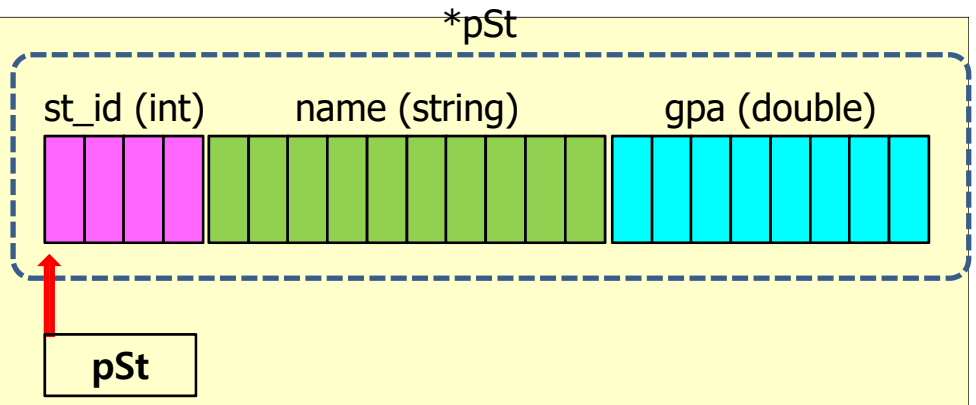
```
Initial state
0  Name: Mercury  Rel Mass:   0.056 Dist from Sun:   57.9
1  Name: Venus    Rel Mass:   0.815 Dist from Sun:  108.0
2  Name: Earth    Rel Mass:   1.000 Dist from Sun:  150.0
3  Name: Mars     Rel Mass:   0.107 Dist from Sun:  228.0
4  Name: Jupiter  Rel Mass: 318.000 Dist from Sun:  778.0
5  Name: Saturn   Rel Mass:  95.100 Dist from Sun: 1430.0
6  Name: Uranus   Rel Mass:  14.500 Dist from Sun: 2870.0
7  Name: Neptune  Rel Mass:  17.200 Dist from Sun: 4500.0
8  Name: Pluto    Rel Mass:   0.110 Dist from Sun: 5900.0

After sorting by relative mass:
0  Name: Mercury  Rel Mass:   0.056 Di
1  Name: Mars     Rel Mass:   0.107 Di
2  Name: Pluto    Rel Mass:   0.110 Di
3  Name: Venus    Rel Mass:   0.815 Di
4  Name: Earth    Rel Mass:   1.000 Di
5  Name: Uranus   Rel Mass:  14.500 Di
6  Name: Neptune  Rel Mass:  17.200 Di
7  Name: Saturn   Rel Mass:  95.100 Di
8  Name: Jupiter  Rel Mass: 318.000 Di

After sorting by distance from sun:
0  Name: Mercury  Rel Mass:   0.056 Dist from Sun:   57.9
1  Name: Venus    Rel Mass:   0.815 Dist from Sun:  108.0
2  Name: Earth    Rel Mass:   1.000 Dist from Sun:  150.0
3  Name: Mars     Rel Mass:   0.107 Dist from Sun:  228.0
4  Name: Jupiter  Rel Mass: 318.000 Dist from Sun:  778.0
5  Name: Saturn   Rel Mass:  95.100 Dist from Sun: 1430.0
6  Name: Uranus   Rel Mass:  14.500 Dist from Sun: 2870.0
7  Name: Neptune  Rel Mass:  17.200 Dist from Sun: 4500.0
8  Name: Pluto    Rel Mass:   0.110 Dist from Sun: 5900.0

After sorting by name using strcmp and strcpy:
0  Name: Earth    Rel Mass:   1.000 Dist from Sun:  150.0
1  Name: Jupiter  Rel Mass: 318.000 Dist from Sun:  778.0
2  Name: Mars     Rel Mass:   0.107 Dist from Sun:  228.0
3  Name: Mercury  Rel Mass:   0.056 Dist from Sun:   57.9
4  Name: Neptune  Rel Mass:  17.200 Dist from Sun: 4500.0
5  Name: Pluto    Rel Mass:   0.110 Dist from Sun: 5900.0
6  Name: Saturn   Rel Mass:  95.100 Dist from Sun: 1430.0
7  Name: Uranus   Rel Mass:  14.500 Dist from Sun: 2870.0
8  Name: Venus    Rel Mass:   0.815 Dist from Sun:  108.0
```

구조체와 포인터

구조체 변수를 가리키는 포인터

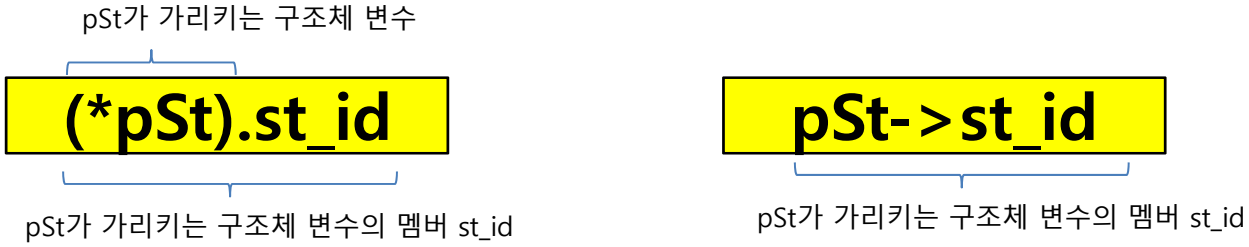
◆ 구조체 변수를 가리키는 포인터

```
typedef struct {
    int st_id;
    char name[20];
    double gpa;
} Student;

Student *pSt;

Student s = { 20070001, "홍길동", 4.3 };

pSt = &s;

printf("학번=%d 이름=%s 학점=%f \n", s.st_id, s.name, s.gpa);
printf("학번=%d 이름=%s 학점=%f \n", (*pSt).st_id,
     (*pSt).name,  (*pSt).gpa);
```



Arrow (->) Operator, 포인터를 사용한 멤버 접근

◆ Arrow Operator (-> 연산자)는 구조체 포인터로 구조체 멤버를 참조할 때 사용




```
Student *pSt;

Student s = { 20070001, "홍길동", 180.2 };

pSt = &s;

printf("학번=%d 이름=%s 성적=%lf \n", s.st_id, s.name, s.gpa);
printf("학번=%d 이름=%s 성적=%lf \n", (*pSt).st_id,(*pSt).name,(*pSt).gpa);
printf("학번=%d 이름=%s 성적=%lf \n", pSt->st_id, pSt->name, pSt->gpa);
```

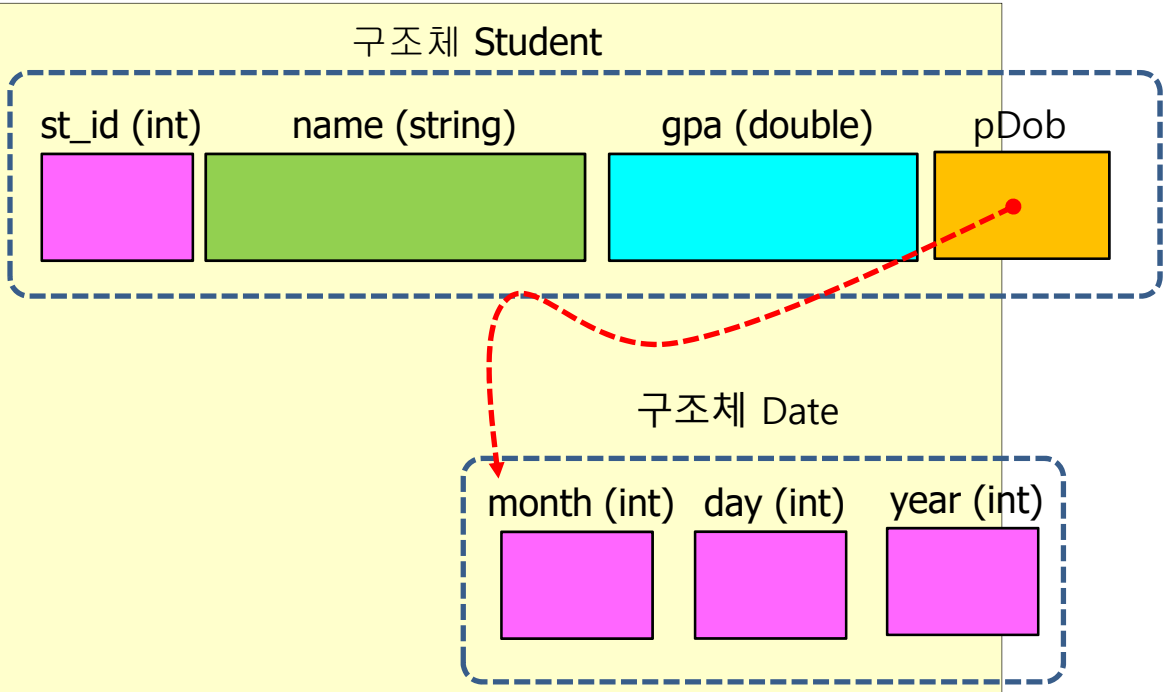
예제

```c
// 포인터를 통한 구조체 참조
#include <stdio.h>

typedef struct {
    int st_id;
    char name[20];
    double gpa;
} Student;

int main(void)
{
    Student s = { 20070001, "홍길동", 4.3};
    Student *pSt;

    pSt = &s;

    printf("학번=%d 이름=%s 키=%f \n", s.st_id, s.name, s.gpa);
    printf("학번=%d 이름=%s 키=%f \n", (*pSt).st_id,(*pSt).name,
        (*pSt).gpa);
    printf("학번=%d 이름=%s 키=%f \n", pSt->st_id, pSt->name,
        pSt->gpa);

    return 0;
}
```

```
학번=20070001 이름=홍길동 학점=4.300000
학번=20070001 이름=홍길동 학점=4.300000
학번=20070001 이름=홍길동 학점=4.300000
```

포인터를 멤버로 가지는 구조체

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

typedef struct
{
    int st_id;
    char name[20];
    double gpa;
    Date *pDob; // Date of birth
} Student;
```



포인터를 멤버로 가지는 구조체

```
int main(void)
{
    Date d = { 3, 20, 1980 };
    Student s = { 20070001, "Kim", 4.3 };

    s.pDob = &d;

    printf("학번: %d\n", s.st_id);
    printf("이름: %s\n", s.name);
    printf("학점: %f\n", s.gpa);
    printf("생년월일: %d년 %d월 %d일\n", s.pDob->year,
          s.pDob->month, s.pDob->day);
    return 0;
}
```

```
학번: 20070001
이름: Kim
학점: 4.300000
생년월일: 1980년 3월 20일
```

구조체 포인터를 함수 호출에서 인수 (argument)로 전달 사용

구조체와 함수

◆ 구조체를 함수의 인수로 전달하는 경우

- 구조체의 **복사본**이 함수로 전달되게 된다.
- 만약 구조체의 크기가 크면 그만큼 긴 시간과 큰 메모리가 소요된다.

```
int equal(Student s1, Student s2)
{
    if( strcmp(s1.name, s2.name) == 0 )
        return 1;
    else
        return 0;
}
```

구조체와 함수

◆ 구조체의 포인터를 함수의 인수로 전달하는 경우

- 시간과 공간을 절약할 수 있다.
- 원본 훼손의 가능성이 있다 => const로 설정하여 변경 방지

```
int equal(Student const *pSt1, Student const *pSt2)
{
    if( strcmp(pSt1->name, pSt2->name) == 0 )
        return 1;
    else
        return 0;
}
```

포인터를 통한 구조체의 변경을 막는다.

구조체 포인터를 함수의 인수로 전달하는 경우

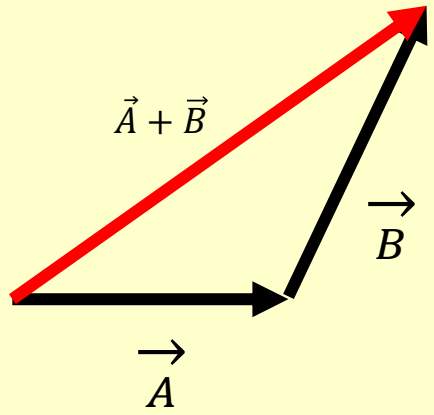
◆ 구조체 포인터를 인수로 전달

- 구조체 포인터를 인수로 전달받은 후,
  표준 입력 장치로 부터 세부 데이터를 입력 받아 구조체의 각 속성 값으로 설정
- 구조체 포인터를 통하여 해당 구조체의 데이터를 직접 변경

```
void init_Student(Student *pSt)
{
    printf("나이:");
    scanf("%d", &pSt->age);
    printf("이름:");
    scanf("%s", pSt-> name);
    printf("성적:");
    scanf("%lf", &pSt->gpa);
}
```

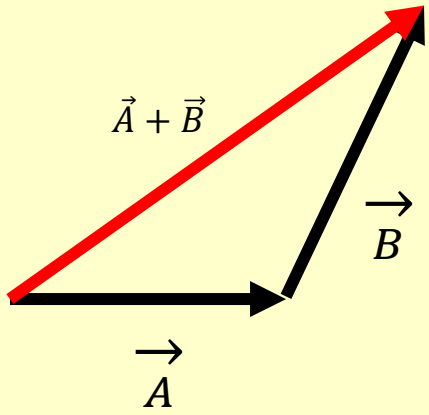
구조체 포인터 인수 전달 예제 (1)

```c
#include <stdio.h>

typedef struct {
    double x;
    double y;
} Vector;

Vector vector_sum(Vector a, Vector b);
Vector vector_sum_pointers(Vector *pA, Vector *pB);

int main(void)
{
    Vector a = { 2.0, 3.0 };
    Vector b = { 5.0, 6.0 };
    Vector sum;

    //sum = vector_sum(a, b);
     sum = vector_sum_pointers(&a, &b);
    printf("벡터의 합은 (%lf, %lf)입니다.\n", sum.x, sum.y);

    return 0;
}
```



구조체 포인터 인수 전달 예제 (2)

```
Vector vector_sum(Vector a, Vector b)
{
     Vector result;

    result.x = a.x + b.x;
    result.y = a.y + b.y;

    return result;
}

Vector vector_sum_pointers(Vector *pA, Vector *pB)
{
     Vector result;

    result.x = pA->x + pB->x;
    result.y = pA->y + pB->y;

    return result;
}
```



벡터의 합은 (7.000000, 9.000000)입니다.

구조체 배열 용용 프로그램 (1)

◆ 학과 내에서 가장 평점이 높은 학생을 찾아서 학생의 이름과 학번, 평점을 화면에 출력하는 프로그램 작성

```
/* Processing Student Records (1) */
#include <stdio.h>
typedef struct {
    int st_id;
    char name[20];
    double gpa;
} Student;

int main(void)
{
   Student st_array[] =
   {    {20120001, "홍길동", 4.2}, {20120002, "김철수", 3.2}, {20120002, "김영희", 3.9}
   };
```

구조체 배열 용용 프로그램 (2)

```
/* Processing Student Records (2) */

  Student *pBest_grade_st;
  int i, size;

  size = sizeof(st_array)/sizeof(st_array[0]);
  pBest_grade_st = &st_array[0];

  for(i=1; i< size; i++)
  {
        if( st_array[i].gpa > pBest_grade_st->gpa )
              pBest_grade_st = &st_array[i];
  }
  printf("평점이 가장 높은 학생은 (이름 %s, 학번 %d, 평점 %lf)입니다\n",
        pBest_grade_st->name, pBest_grade_st->st_id,
        pBest_grade_st->gpa);
}
```

바이트 저장 순서 (Byte Ordering)
- Little Endian
- Big Endian

Byte Ordering (바이트 저장 순서)

◆ 바이트 저장 순서

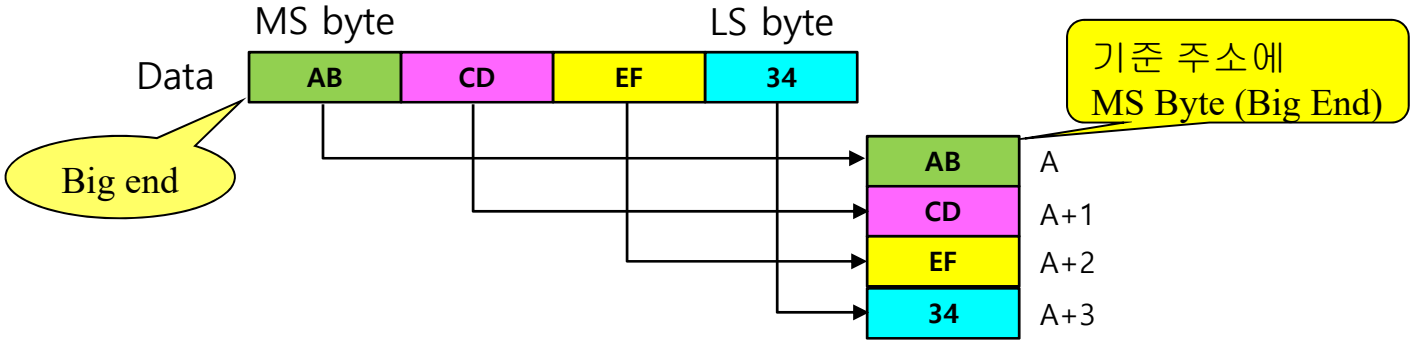
- 컴퓨터에서 사용되는 CPU 종류에 따라 여러 바이트로 표현되는 데이터 (예: 2바이트 크기의 short, 4바이트 크기의 정수 (integer), 8바이트 크기의 double)를 메모리에 저장할 때, 바이트 저장 순서가 서로 다름
- Big Endian 체계: (기준이 되는) 시작 주소에 MSB (most significant byte)인 Big End가 저장됨
- Little Endian 체계: (기준이 되는) 시작 주소에 LSB (least significant byte)인 Little End 가 저장됨

◆ 프로그래밍에서 바이트 순서를 고려하여야 하는 이유

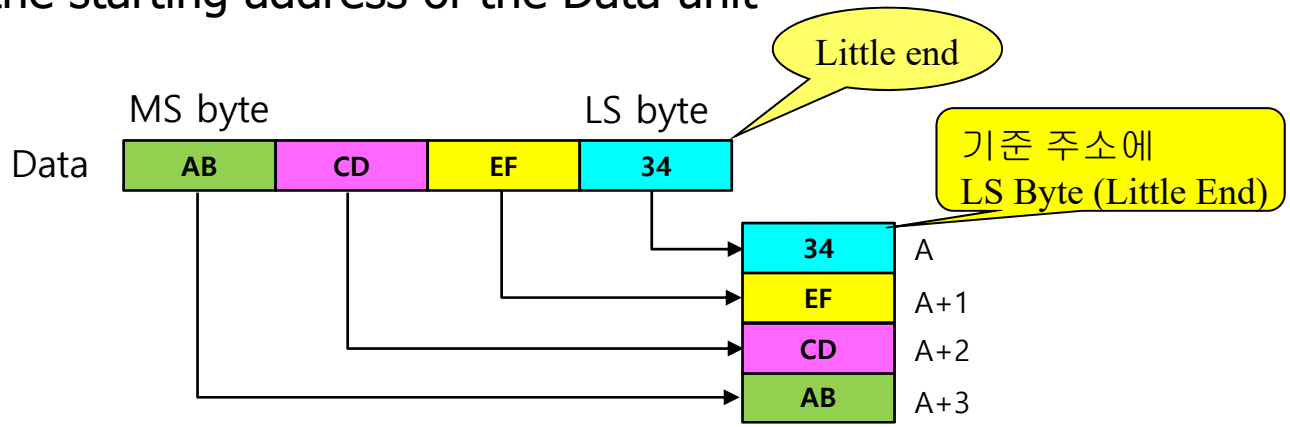
- 하나의 컴퓨터 내에서만 데이터를 저장하고, 읽어 사용하는 경우 문제가 없음
- 인터넷을 통하여 다른 컴퓨터로 데이터를 전달하는 경우, 각 송신측과 수신측 컴퓨터의 바이트 저장 순서 체계가 다른 경우, 문제가 발생됨
- 해결방안: 인터넷으로 전송할 때, 항상 Big Endian 체계로 변환시켜 전송함.

◆ Big-endian vs. Little-endian byte ordering

- big-endian system stores the most significant byte (big end) of Data in the starting address of the Data unit

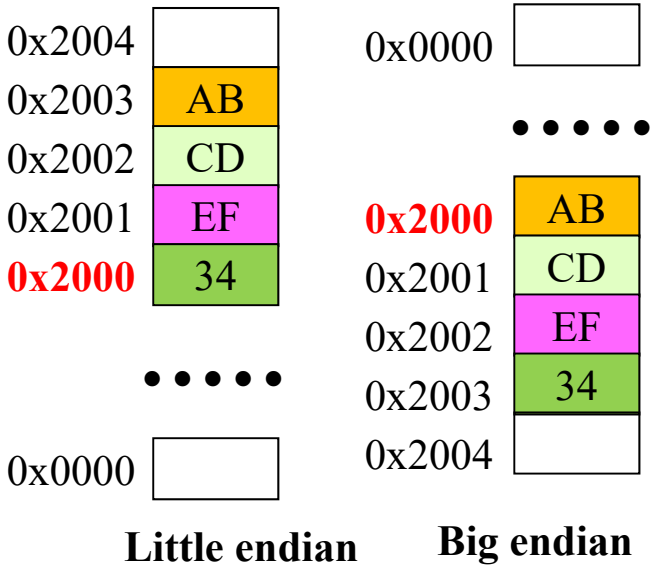


- little-endian system stores the least significant byte (little end) of Data in the starting address of the Data unit

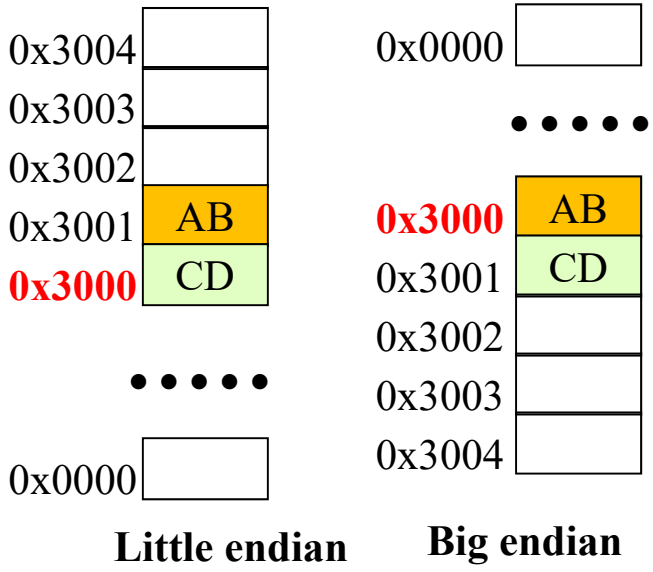


◆ Byte ordering example

```
unsigned long *pL;
pL = (unsigned long *)0x2000;
*pL = 0xABCDEF34;
```



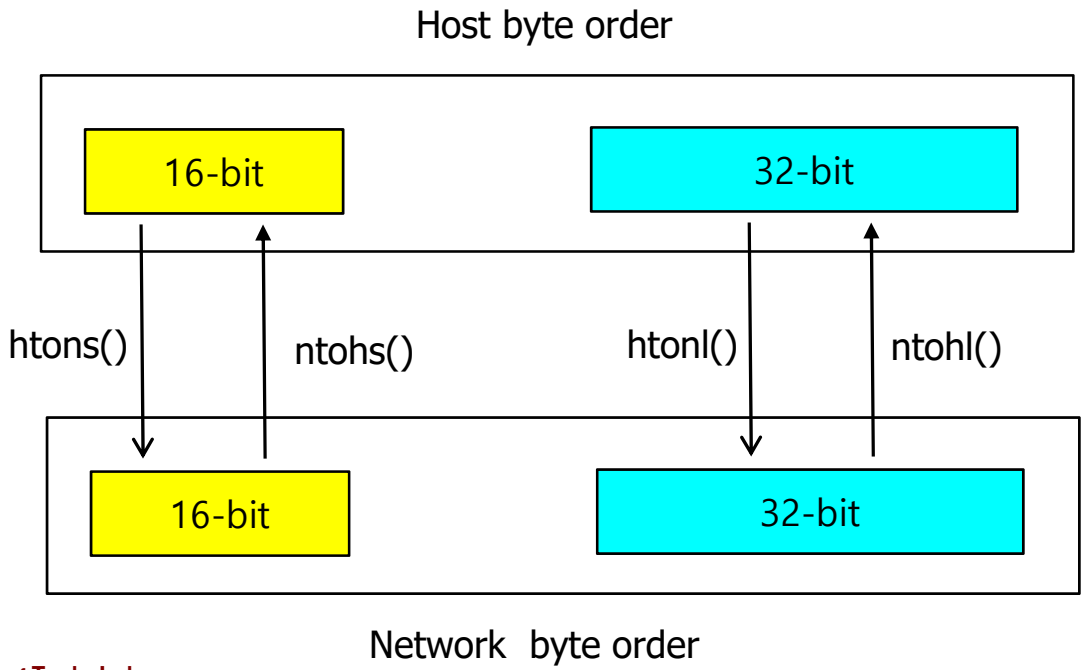

```
unsigned short *pS;
pS = (unsigned short *)0x3000;
*pS = 0xABCD;
```



◆ Byte-order transformation in Internet Socket Programs

- Standard of Internet byte order: Big Endian
- Byte-ordering 관련 라이브러리 함수들:

```
uint16_t  htons (uint16_t  host_short);
uint16_t ntohs (uint16_t  network_short);
uint32_t  htonl (uint32_t  host_long);
uint32_t ntohl (uint32_t  network_long);
```



Test Byte Ordering

```
/* Test Byte Ordering */
#include <stdio.h>
void printHexChar(unsigned char uchar);

int main(void)
{
    union {
       unsigned long ul;
       unsigned char uc[4];
    } un;
    unsigned long ul2;

    un.ul = 0xABCDEF34;
    printf("Address of ul(0xABCDEF34) = %p\n", &un.ul);
    printf("Unsigned long ul (0xABCDEF34) in byte order : ");
    for (int i = 0; i < 4; i++)
    {
       printf("%p(", &un.uc[i]);
       printHexChar(un.uc[i]);
       printf(") ");
    }
    printf("\n");
}
```

Test Byte Ordering

```
/* Testi Byte Ordering (2) */
void printHexChar(unsigned char uchar)
{
   char hexChar[17] = "0123456789ABCDEF";
   unsigned char ch;

   ch = hexChar[(uchar & 0xF0) >> 4];
   printf("%c", ch);
   ch = hexChar[(uchar & 0x0F)];
   printf("%c", ch);
}
```

```
Address of ul(0xABCDEF34) = 0036FC2C
Unsigned long ul (0xABCDEF34) in byte order : 0036FC2C(34) 0036FC2D(EF) 0036FC2E(CD) 0036FC2F(AB)
```

Byte Ordering 변환

```
unsigned long byteReorderLong(unsigned long ul)
{
    unsigned long res;

    res = 0;
    res += (ul & 0xFF000000) >> 24;
    res += (ul & 0x00FF0000) >> 8;
    res += (ul & 0x0000FF00) << 8;
    res += (ul & 0x000000FF) << 24;

    return res;
}

unsigned long byteReorderShort(unsigned short us)
{
    unsigned short res;

    res = 0;
    res += (us & 0xFF00) >> 8;
    res += (us & 0x00FF) << 8;

    return res;
}
```

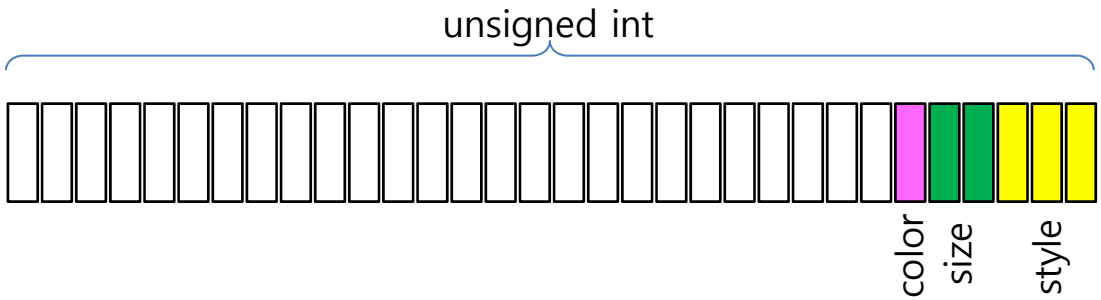
비트 단위 구조체 (Bit-Field Structure)

비트 단위 구조체 (bit-field structure)

◆ struct with bit-unit members

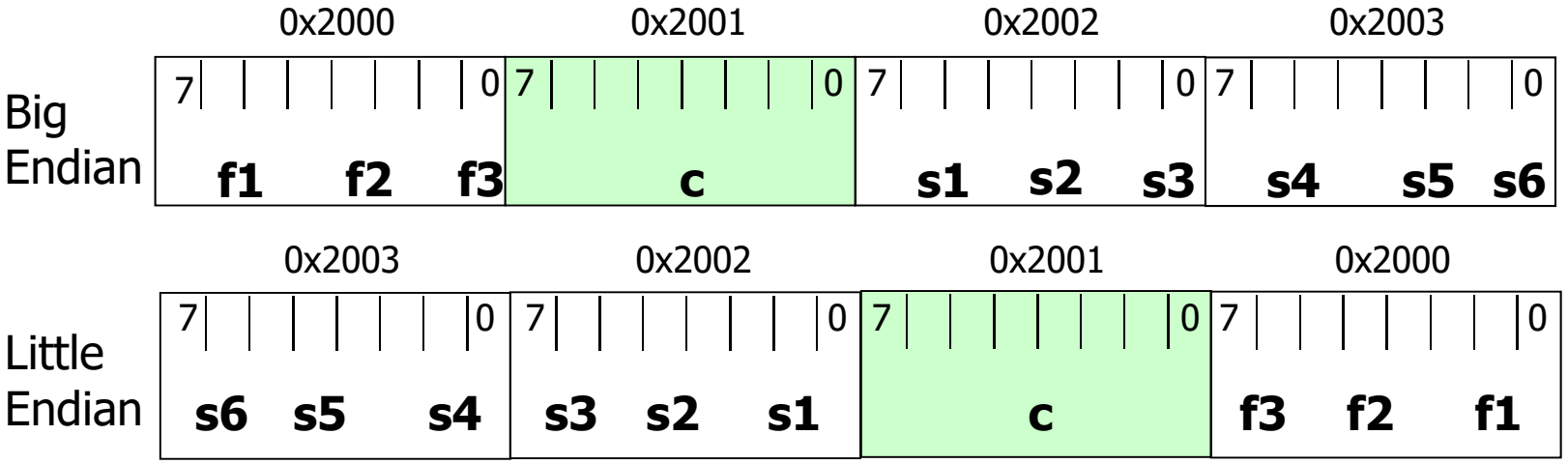
```
typedef struct {
    DataType member_1: number_bits;
    DataType member_2: number_bits;
    ...
} TagName;
```

```
typedef struct {
    unsigned style : 3;
    unsigned size  : 2;
    unsigned color : 1;
} Product ;
```



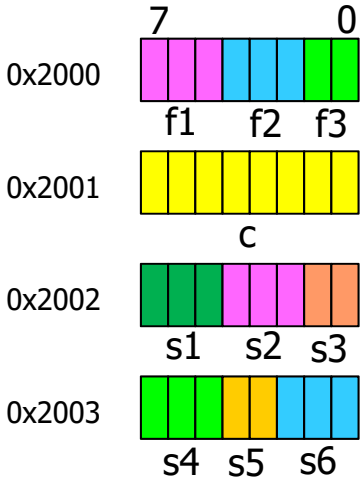
◆ Field Ordering Example

```
typedef struct {
    char f1:3,
         f2:3,
         f3:2;
    char c;
    short s1:3,
          s2:3,
          s3:2,
          s4:3,
          s5:2,
          s6:3;
} BFU; // Bit-field Unit
BFU *pBFU =(BFU *)0x2000;
```



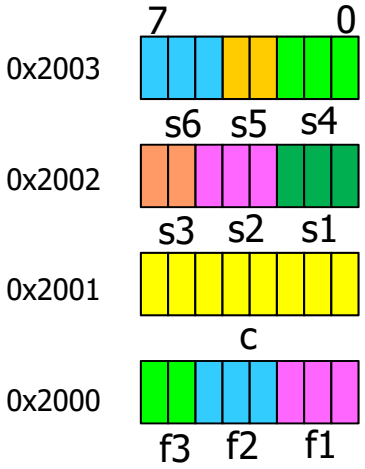
Example of Bit-field mapping

◆ Bit-field mapping in Little-Endian and Big-Endian



Big-Endian Byte Ordering,
Bit-field Mapping
(Apple Mac)

```
typedef struct {
   unsigned char f1:3,
      f2:3,
      f3:2;
   char c;
   unsigned short s1:3,
      s2:3,
      s3:2,
      s4:3,
      s5:2,
      s6:3;
} BFU_32;
BFU_32 *pBFU32 =
   (BFU_32 *)0x2000;
```



Little-Endian Byte Ordering,
Bit-field Mapping
(PC Intel CPU)

Testing Bit-Field Structure

```
/* Testing Little Endian, Big Endian, Bit-Field Struct  (1) */
#include <stdio.h>

#define GET_BIT(w, k) (((w) >> (k)) & 0x01)
#define SET_BIT(w, k) ((w) |= (0x01 << (k)))

typedef struct {
   unsigned char
      first : 4,
      second : 4;
} TwoFieldBitStruct;

typedef struct {
   unsigned char
      style : 3,
      size : 2,
      color : 3;
} ThreeFieldsBitStruct;

void printBits(unsigned char *pU_char);
```

Testing Bit-Field Structure

```
/* Testing Little Endian, Big Endian, Bit-Field Struct  (2) */

int main(void)
{
    TwoFieldBitStruct octet1;
    ThreeFieldsBitStruct octet2;
    unsigned char *pU_char;

    octet1.first = 6;  // 0110
    octet1.second = 3; // 0011
    printf("sizeof(octet1)=%d\n", sizeof(octet1));
    printf("octet1 is defined as {first (4-bit) =%d, second (4-bit)=%d}\n",
        octet1.first, octet1.second);
    printf("octet1=%#X ", octet1);
    pU_char = (unsigned char *)&octet1;
    printBits(pU_char);

    octet2.style = 5;  // 101
    octet2.size = 2;   // 10
    octet2.color = 6;  // 110
    printf("sizeof(octet2)=%d\n", sizeof(octet2));
    printf("octet2 is defined as {style(3-bit)=%d, size (2-bit)=%d,
        color (3-bit)=%d}\n", octet2.style, octet2.size, octet2.color);
    printf("octet2=%#X ", octet2);
    pU_char = (unsigned char *)&octet2;
    printBits(pU_char);
    return 0;
}
```

Testing Bit-Field Structure

```
/* Testing Little Endian, Big Endian, Bit-Field Struct  (3) */

void printBits(unsigned char *pU_char)
{
    unsigned char byte;

    byte = *pU_char;

    printf(", in bit pattern (");
    for (int i = 7; i >= 0; i--)
    {
       printf("%d", GET_BIT(byte, i);
    }
    printf(")\n");
}
```

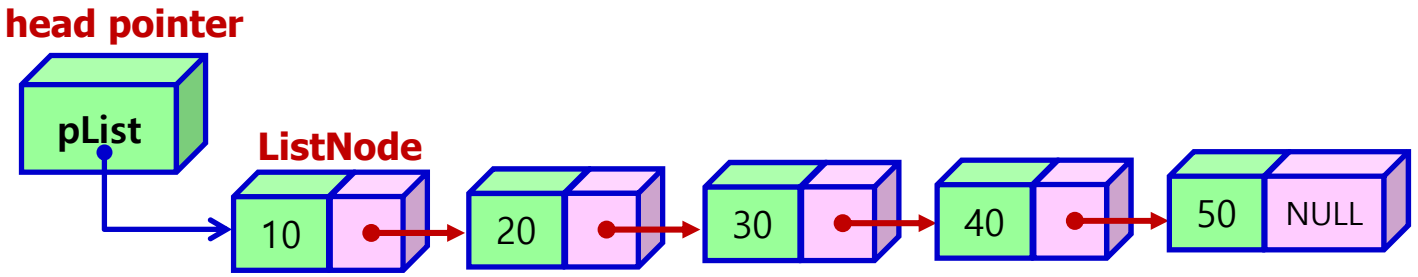
```
sizeof(octet1)=1
octet1 is defined as {first (4-bit) =6, second (4-bit)=3}
octet1=0X36 , in bit pattern (00110110)
sizeof(octet2)=1
octet2 is defined as {style(3-bit)=5, size (2-bit)=2, color (3-bit)=6}
octet2=0XD5 , in bit pattern (11010101)
계속하려면 아무 키나 누르십시오 . . .
```

자기 참조 구조체 개요
(Self-referential structure)

자기참조 구조체 (Self-referential Structure)

◆ Self-referential structure(자기참조 구조체)

- a special structure that includes a pointer which points the same type structure

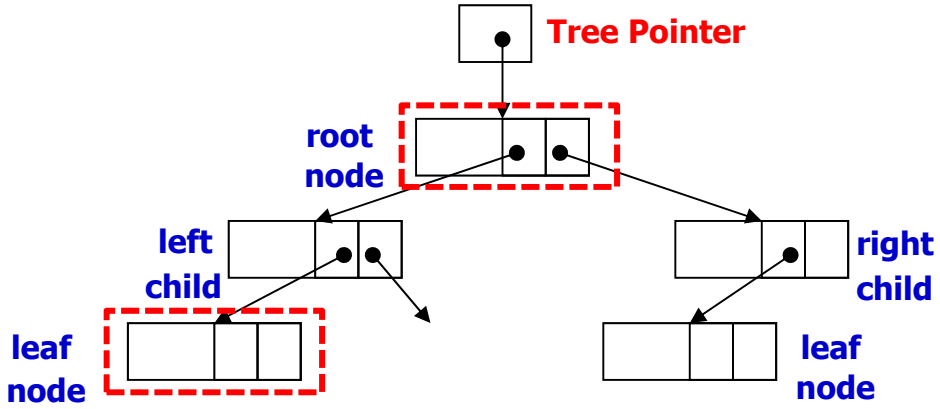



```
// definition of Data field
typedef struct {
    int id;
    char name[20];
    char phone[12];
} Data;

// definition of node
typedef struct ListNode {
    Data *pData;  // Data field
    struct ListNode *next;  // pointer to next node
} ListNode;
```

자기참조 구조체의 예 – Binary Search Tree Node

◆Binary Tree Node

- A node in a binary tree is like a node in a linked list, except it has two node pointer fields:




```
typedef struct TreeNode
{
    Data *pData; // Data field
    struct TreeNode *left;   // link field to left child
    struct TreeNode *right;  // link field to right child
} TreeNode;
```

Homework 9

Homework 9

9.1 구조체를 사용하여 함수에 인수를 전달하고, 결과값을 반환시키는 기능을 예를 들어 설명하라.

9.2 Byte Ordering이란 무엇인가에 대하여 설명하고, 인터넷을 통한 정보 전송에서 바이트 저장순서를 고려하여야 하는 이유는 무엇인가 ?

9.3 비트 필드 구조체란 무엇인가, 그리고 왜 필요한가에 대하여 설명하고, Big Endian CPU와 Little Endian CPU가 각각 저장하는 차이점을 예를 들어 설명하라.

Homework 9

9.4 복소수 계산을 위한 구조체 정의 (Struct Cmplx)

1) 복소수는 실수부와 허수부를 가지며, c = real (실수부) + j imaginary(허수부) 로 표현된다. 복소수를 위한 구조체 struct Cmplx를 작성하라. 실수부와 허수부는 보다 높은 정밀도를 위하여 double 데이터 유형을 사용한다. 또한, 복소수를 "rrr.ddd + j iii.ddd (magnitude mmm.ddd)" 양식으로 출력하는 함수 void printCmplx(const Cmplx c)를 작성하라.

```
typedef struct
{
    double real;
    double imaginary;
} Cmplx;
```

2) 복소수의 4칙 연산을 위한 함수를 정의하라.

- Cmplx cmplxAdd(const Cmplx c1, const Cmplx c2);
- Cmplx cmplxSubtract(const Cmplx c1, const Cmplx c2);
- Cmplx cmplxMultiply(const Cmplx c1, const Cmplx c2);
- Cmplx cmplxDivide(const Cmplx c1, const Cmplx c2);

3) 복소수 배열 (예: Cmplx cmplxs[7])을 정의하라.

4) 표준입력장치로 부터 4개의 double형 데이터를 입력받아, 실수부와 허수부의 값으로 설정하고, 복소수를 반환하는 함수 Cmplx inputCmplx(void)를 작성하라. 입력 포멧은 "rrr.ddd iii.ddd"를 사용하라. 이함수를 사용하여 복소수 배열의 첫번째 두개의 복소수(cmplxs[0], cmplxs[1])의 값을 차례로 입력하도록 하라.

5) 위에서 구현한 4개의 복소수 연산 함수를 사용하여, 표준 입력장치로 부터 입력을 받아 설정된 두 개의 복소수 (cmplxs[0], cmplxs[1])의 addition, subtraction, multiplication 과 division을 계산하고, 그 결과를 각각 cmplxs[2], cmplxs[3], cmplxs[4], cmplxs[5]에 저장하라.
cmplxs[1]과 cmplxs[5]의 곱셈 계산을 하여 cmplxs[6]에 저장하고, 이 복소수가 cmplxs[0]과 동일한 것을 확인하라.

6) 복소수 배열에 포함된 각 복소수 (cmplxs[0] ~ cmplxs[6])를 "rrr.ddd + j iii.ddd (magnitude mmm.ddd)" 양식으로 한줄에 복소수 1개씩을 출력하라.

7) 인수로 전달된 2개의 복소수 크기를 비교하는 함수 double compareCmplx(const Cmplx c1, const Cmplx c2)를 작성하라. 복소수의 비교는 복소수의 크기 (magnitude = sqrt (real^2 + imaginary^2))에 따라 결정하며, c1의 크기 – c2의 크기를 계산하여 결과값을 반환하도록 하라.

8) 복소수 배열의 신속한 정렬을 위하여, 퀵정렬 함수 (quickSortCmplx(Cmplx cmplxs[], int size))를 작성하라. 이 복소수 배열의 퀵정렬함수는 복소수 배열을 파라메터로 전달 받으며, 복소수 크기 (magnitude)값의 오름차순으로 정렬한다.

9) 정렬된 복소수 배열을 "rrr.ddd + j iii.ddd (magnitude mmm.ddd)" 양식으로 한줄에 복소수 1개씩을 출력하라.

◆ Cmplx.h (예시)

```
/* Cmplx.h */
#ifndef CMPLX_H
#define CMPLX_H

typedef struct
{
      double real;
      double imag;  // imaginary
} Cmplx;

Cmplx inputCmplx();
void printCmplx(const Cmplx c);
Cmplx cmplxAdd(const Cmplx c1, const Cmplx c2);
Cmplx cmplxSub(const Cmplx c1, const Cmplx c2);
Cmplx cmplxMul(const Cmplx c1, const Cmplx c2);
Cmplx cmplxDiv(const Cmplx c1, const Cmplx c2);
void sortCmplx(Cmplx cmplxs[], int size);

#endif
```

◆ main() 함수 (예시)

```
/* main_cmplx.cpp  (1) */

#include <stdio.h>
#include "Cmplx.h"

void main()
{
        Cmplx cmplxs[7];

        cmplxs[0] = inputCmplx();
        cmplxs[1] = inputCmplx();
        printf("cmplxs[0] = "); printCmplx(cmplxs[0]); printf("\n");
        printf("cmplxs[1] = "); printCmplx(cmplxs[1]); printf("\n");
        cmplxs[2] = cmplxAdd(cmplxs[0], cmplxs[1]);
        printf("cmplxs[2] = cmplxs[0] + cmplxs[1] = \n  ");
        printCmplx(cmplxs[0]); printf(" + "); printCmplx(cmplxs[1]); printf(" = "); printCmplx(cmplxs[2]); printf("\n");
        cmplxs[3] = cmplxSub(cmplxs[0], cmplxs[1]);
        printf("cmplxs[3] = cmplxs[0] - cmplxs[1] = \n  ");
        printCmplx(cmplxs[0]); printf(" - "); printCmplx(cmplxs[1]); printf(" = "); printCmplx(cmplxs[3]); printf("\n");
        cmplxs[4] = cmplxMul(cmplxs[0], cmplxs[1]);
        printf("cmplxs[4] = cmplxs[0] * cmplxs[1] = \n  ");
        printCmplx(cmplxs[0]); printf(" * "); printCmplx(cmplxs[1]); printf(" = "); printCmplx(cmplxs[4]); printf("\n");
        cmplxs[5] = cmplxDiv(cmplxs[0], cmplxs[1]);
        printf("cmplxs[5] = cmplxs[0] / cmplxs[1] = \n  ");
        printCmplx(cmplxs[0]); printf(" / "); printCmplx(cmplxs[1]); printf(" = "); printCmplx(cmplxs[5]); printf("\n");
        cmplxs[6] = cmplxMul(cmplxs[1], cmplxs[5]);
        printf("cmplxs[6] = cmplxs[1] * cmplxs[5] = \n  ");
        printCmplx(cmplxs[1]); printf(" * "); printCmplx(cmplxs[5]); printf(" = "); printCmplx(cmplxs[6]); printf("\n");
```

◆ main() 함수 (예시)

```
/* main_cmplx.cpp  (2) */


        printf("Before sorting complexs: \n");
        for (int i = 0; i < 7; i++)
        {
                printf("cmplxs[%d] = ", i);  printCmplx(cmplxs[i]);
                printf("\n");
        }

        sortCmplx(cmplxs, 7);
        printf("Sorted complexs: \n");
        for (int i = 0; i < 7; i++)
        {
                printf("cmplxs[%d] = ", i);  printCmplx(cmplxs[i]);
                printf("\n");
        }
}
```

◆ 실행 결과 (예시)

```
Input complex number (rrr.ddd iii.ddd) : 3.3 4.4
Input complex number (rrr.ddd iii.ddd) : 1.1 2.2
cmplxs[0] = (3.300 + j4.400 (magnitude 5.500))
cmplxs[1] = (1.100 + j2.200 (magnitude 2.460))
cmplxs[2] = cmplxs[0] + cmplxs[1] =
  (3.300 + j4.400 (magnitude 5.500)) + (1.100 + j2.200 (magnitude 2.460)) = (4.400 + j6.600 (magnitude 7.932))
cmplxs[3] = cmplxs[0] - cmplxs[1] =
  (3.300 + j4.400 (magnitude 5.500)) - (1.100 + j2.200 (magnitude 2.460)) = (2.200 + j2.200 (magnitude 3.111))
cmplxs[4] = cmplxs[0] * cmplxs[1] =
  (3.300 + j4.400 (magnitude 5.500)) * (1.100 + j2.200 (magnitude 2.460)) = (-6.050 + j12.100 (magnitude 13.528))
cmplxs[5] = cmplxs[0] / cmplxs[1] =
  (3.300 + j4.400 (magnitude 5.500)) / (1.100 + j2.200 (magnitude 2.460)) = (2.200 - j0.400 (magnitude 2.236))
cmplxs[6] = cmplxs[1] * cmplxs[5] =
  (1.100 + j2.200 (magnitude 2.460)) * (2.200 - j0.400 (magnitude 2.236)) = (3.300 + j4.400 (magnitude 5.500))
Before sorting complexs:
cmplxs[0] = (3.300 + j4.400 (magnitude 5.500))
cmplxs[1] = (1.100 + j2.200 (magnitude 2.460))
cmplxs[2] = (4.400 + j6.600 (magnitude 7.932))
cmplxs[3] = (2.200 + j2.200 (magnitude 3.111))
cmplxs[4] = (-6.050 + j12.100 (magnitude 13.528))
cmplxs[5] = (2.200 - j0.400 (magnitude 2.236))
cmplxs[6] = (3.300 + j4.400 (magnitude 5.500))
Sorted complexs:
cmplxs[0] = (2.200 - j0.400 (magnitude 2.236))
cmplxs[1] = (1.100 + j2.200 (magnitude 2.460))
cmplxs[2] = (2.200 + j2.200 (magnitude 3.111))
cmplxs[3] = (3.300 + j4.400 (magnitude 5.500))
cmplxs[4] = (3.300 + j4.400 (magnitude 5.500))
cmplxs[5] = (4.400 + j6.600 (magnitude 7.932))
cmplxs[6] = (-6.050 + j12.100 (magnitude 13.528))
```

9.5 구조체 Time을 사용하는 Clock system

1) 시간을 나타내기 위하여, hour, minute, 및 second를 포함하는 구조체 Time을 작성하라.

```
typedef struct {
   int      hour;
   int      min;
   int      sec;
} Time;
```

2) 표준 입력 장치로 부터 3개의 정수를 입력 받아, 이를 각각 시, 분, 초의 값으로 설정하는 구조체 Time의 변수를 생성하여 반환하는 Time inputTime(void) 함수를 구현하라. 예를 들어, "10 5 30" 로 입력된 정수는 구조체 Time 변수의 "10시 5분 30초"로 설정된다. 시간을 나타내는 값은 0시 ~ 23시의 값을 가지도록 할 것.

3) 구조체 Time 변수의 값을 "hh:mm:ss" 양식으로 출력하는 함수 printTime(const Time *tptr1)을 작성하라. 구조제 Time 변수는 "00:00:00" ~ "23:59:59" 구간의 값을 가진다.

4) 구조체 Time 변수 값을 지정된 초단위로 증가하는 함수 "void incrementTime(Time *, int)"를 작성하라.

5) main() 함수에서 inputTime() 함수를 사용하여 시간을 입력 받고, 초단위 정수 값 seconds를 입력 받은 후, 구조체 Time 변수의 값을 증가시킨 후, 그 결과를 출력하라. 분과 시는 60초와 60분 단위로 증가되어야 한다. 예를 들어, "23:59:59" 에 2 초를 더하면 "00:00:01"이 되어야 한다.

9.5 (계속)

6) 두개의 구조체 Time 변수를 비교하는 "int compareTime(const Time *tptr1, const Time *tptr2)" 함수를 작성하라. 이 함수는 call-by-pointer로 (구조체 포인터)로 전달되는 두개의 시간을 비교하여, 그 차이 (초 단위)를 반환한다. 반환 데이터 유형은 정수이며, 초단위 값을 나타낸다. 위 2)에서 초기 설정된 시간 값과 4)에서 증가된 시간값을 비교하여, 그 차이를 compareTime() 함수를 사용하여 계산하고, 그 결과를 반환하여, 정확한 계산이 이루어 졌는가를 확인하라.

7) 5개의 구조체 time 배열 times[]을 선언하고, 초기값으로 {{23, 59, 59}, {9, 0, 5}, {13, 30, 0}, {3, 59, 59}, {0, 0, 0}}으로 설정하라.

8) 구조체 time 배열 times[]를 선택정렬 방식으로 정렬하는 함수 selectSortTime(Time *tptr1, int size)을 작성하라.

9) 위에서 구현한 inputTime(), printTime(), incrementTime(), compareTime(), selectSortTime() 함수를 사용하여 해당 기능을 시험할 수 있도록 main() 함수를 작성하고, 실행결과를 확인하라.

◆ Time.h (예시)

```
/* Time.h */

#ifndef TIME_H
#define TIME_H

typedef struct
{
      int hour;
      int min;
      int sec;
} Time;

Time inputTime();
Time initTime(int h, int m, int s);
void printTime(const Time* t);
void incrementTime(Time* t, int incr_sec);
int elapsedSeconds(const Time* t);
int compareTime(const Time* t1, const Time* t2);
void selectSortTime(Time* times, int size);

#endif
```

◆ main() 함수 (예시)

```
/* main_time.cpp (1) */

#include <stdio.h>
#include "Time.h"
#define NUM_TIMES 5
void main()
{
      Time t1, t2;
      Time times[NUM_TIMES];
      int incr_secs, diff_sec;

      t1 = t2 = inputTime();
      printf("Input time t1 = ");  printTime(&t1); printf("\n");
      printf("input seconds to increment : ");
      scanf("%d", &incr_secs);
      incrementTime(&t2, incr_secs);
      printf("After incrementing %d secs, t2 = ", incr_secs);  printTime(&t2); printf("\n");
      diff_sec = compareTime(&t1, &t2);
      printf("Difference between t1 and t2 is %d secs\n", diff_sec);
```

◆ main() 함수 (예시)

```
input hour minute sec : 23 59 59
Input time t1 = (23:59:59)
input seconds to increment : 2
After incrementing 2 secs, t2 = (00:00:01)
Difference between t1 and t2 is 2 secs

Before sorting times :
times[0] = (23:59:59)
times[1] = (09:00:05)
times[2] = (13:30:00)
times[3] = (03:59:59)
times[4] = (00:00:00)
After selection sorting of times :
times[0] = (00:00:00)
times[1] = (03:59:59)
times[2] = (09:00:05)
times[3] = (13:30:00)
times[4] = (23:59:59)
```

```
/* main_time.cpp (1) */

      times[0] = initTime(23, 59, 59);
      times[1] = initTime(9, 0, 5);
      times[2] = initTime(13, 30, 0);
      times[3] = initTime(3, 59, 59);
      times[4] = initTime(0, 0, 0);
      printf("\nBefore sorting times : \n");
      for (int i = 0; i < NUM_TIMES; i++)
      {
            printf("times[%d] = ", i);  printTime(&times[i]); printf("\n");
      }
      selectSortTime(times, NUM_TIMES);
      printf("After selection sorting of times : \n");
      for (int i = 0; i < NUM_TIMES; i++)
      {
            printf("times[%d] = ", i);  printTime(&times[i]); printf("\n");
      }
}
```