구조체의 포인터

- **구조체의 포인터 변수 선언**

```
typedef struct _date {
    int year;
    int month;
    int day;
} date;
date d1;
date *pd1 = &d1;
```

- **구조체 포인터의 멤버 접근 연산자 ->**

```
pd1->year = 2018;
```

- **접근연산자 ->와 .의 연산자 우선순위 :**
  연산자 중 우선순위가 가장 높음.
  (간접연산자 * 가 우선순위 2위)
  - **연산자 ->와 . : 결합성은 좌에서 우**
  - **연산자 * : 결합성은 우에서 좌**

구조체 포인터의 선언과 사용
(structpointer.c)

```
#include <stdio.h>
struct lecture {
    char name[20];          //강좌명
    int type;//강좌구분 0: 교양, 1: 일반선택, 2: 전공필수, 3: 전공선택
    int credit;             //학점
    int hours;              //시수
};
typedef struct lecture lecture;
char *head[] = { "강좌명", "강좌구분", "학점", "시수" };
char *lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
```

```
[결과]
구조체크기: 32, 포인터크기: 8

    강좌명      강좌구분    학점    시수
    운영체제    전공필수      3       3
 C프로그래밍    전공선택      3       4
           C    전공선택      3       4
```

구조체 포인터의 선언과 사용
(structpointer.c)

```
int main(void) {
    lecture os = { "운영체제", 2, 3, 3 };
    lecture c = { "C프로그래밍", 3, 3, 4 };
    lecture *p = &os;
    printf("구조체크기: %d, 포인터크기: %d₩n₩n", sizeof(os), sizeof(p));
    printf("%10s %12s %6s %6s₩n", head[0], head[1], head[2], head[3]);
    printf("%12s %10s %5d %5d₩n",
        p->name, lectype[p->type], p->credit, p->hours);
    p = &c;
    printf("%12s %10s %5d %5d₩n",
        (*p).name, lectype[(*p).type], (*p).credit, (*p).hours);
    printf("%12c %10s %5d %5d₩n",
        *c.name, lectype[c.type], c.credit, c.hours);
    return 0;
}
```

-> 연산자는 (*). 연산자와 같음!

[결과]
구조체크기: 32, 포인터크기: 8

강좌명	강좌구분	학점	시수
운영체제	전공필수	2	3

공용체 포인터의 선언과 사용 (unionpointer.c)

```
#include <stdio.h>
int main(void) {
    union data {
        char ch;
        int cnt;
        double real;
    };
    typedef union data udata;
    udata value, *p;
    p = &value;
    p->ch = 'a';
    printf("%c %c₩n", p->ch, (*p).ch);
    p->cnt = 100;
    printf("%d ", p->cnt);
    p->real = 3.14;
    printf("%.2f ₩n", p->real);
    return 0;
}
```

공용체 타입을 선언하고 사용

같은 뜻임.

[결과]
a a
100 3.14

구조체의 배열 사용
(structarray.c)

```
#include <stdio.h>
struct lecture {
    char name[20];          //강좌명
    int type;               //강좌구분
    int credit;             //학점
    int hours;              //시수
};
typedef struct lecture lecture;
char *lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
char *head[] = { "강좌명", "강좌구분", "학점", "시수" };
```

[결과]
배열크기: 5

강좌명	강좌구분	학점	시수
인간과 사회	교양	2	2
경제학개론	일반선택	3	3
자료구조	전공필수	3	3
모바일프로그래밍	전공필수	3	4

구조체의 배열 사용
(structarray.c)

```
int main(void)  {
    lecture course[] = { { "인간과 사회", 0, 2, 2 },
    { "경제학개론", 1, 3, 3 },
    { "자료구조", 2, 3, 3 },
    { "모바일프로그래밍", 2, 3, 4 },
    { "고급 C프로그래밍", 3, 3, 4 } };
    int arysize = sizeof(course) / sizeof(course[0]);

    printf("배열크기: %d₩n₩n", arysize);
    printf("%12s    %12s %6s %6s₩n", head[0], head[1], head[2], head[3]);
    printf("=============================================₩n");
    for (int i = 0; i < arysize; i++)
        printf("%16s %10s %5d %5d₩n", course[i].name,
        lectype[course[i].type], course[i].credit, course[i].hours);
    return 0;
}
```

구조체 배열의 초기값 : 중괄호를 중첩하여 사용
외부 중괄호 : 배열 초기화
내부 중괄호 : 배열원소인 구조체 초기화를

[ ] 연산자로써 배열 원소인 구조체를 찾은 후
. 연산자로써 구조체 멤버에 접근.

14. 함수와 포인터 활용

12주차

call by value, call by reference

- 값에 의한 호출(call by value)
  - C 언어의 기본적인 인자 전달 방식
  - 함수 호출 시 실인자의 값이 형식인자에 복사되므로 함수 내에서 형식인자를 수정해도 실인자에는 영향이 없음.

- 참조에 의한 호출 (call by reference)
  - 포인터를 매개변수로 사용하면 실인자와 형식인자가 같은 주소를 갖게 되어 변수의 변화된 값을 공유할 수 있음.

Call by value 사용
(callbyvalue.c)

```
#include <stdio.h>

void increase(int origin, int increment);

int main(void) {
    int amount = 10;
    //amount가 20 증가하지 않음
    increase(amount, 20);
    printf("%d₩n", amount);

    return 0;
}

void  increase(int origin, int increment) {
    origin += increment;
}
```

[결과]
10

amount를 전달 받은 origin 값을 수정했지만 amount에는 영향이 없음.

Call by reference 사용
(callbyreference.c)

```
#include <stdio.h>

void increase(int *origin, int increment);

int main(void) {
    int amount = 10;
    increase( &amount, 20 );
    printf("%d₩n", amount);
    return 0;
}

void increase( int *origin, int increment) {
    *origin += increment;
}
```

[결과]
30

함수로 주소를 전달
-> 함수 내에서 주소를 간접참조하여 값을 수정
-> 호출한 함수에서 변경된 값 확인 가능.

배열을 매개변수로 전달 (arrayparameter.c)

```
#include <stdio.h>
#define ARYSIZE 5
double sum(double g[], int n);
int main(void) {
    double data[] = { 2.3, 3.4, 4.5, 6.7, 9.2 };
    for (int i = 0; i < ARYSIZE; i++)
        printf("%5.1f", data[i]);
    puts("");
    printf("합: %5.1f₩n", sum(data, ARYSIZE));
    return 0;
}
double  sum( double ary[], int n ) {
    double total = 0.0;
    for (int i = 0; i < n; i++)
        total += ary[i];
    return total;
}
```

[결과]
2.3 3.4 4.5 6.7 9.2
합: 26.1

배열 원소 값을 모두 더하는 함수정의 :
- 배열이름은 주소이므로 call by reference 와 같은 의미.
- 배열원소수를 별도의 인자로 전달하기.

다양한 배열원소 참조 방법 (arrayparam.c)

```
#include <stdio.h>
int sumary(int *ary, int SIZE); //int sumary(int ary[], int SIZE)도 가능
int main(void) {
    int point[] = { 95, 88, 76, 54, 85, 33, 65, 78, 99, 82 };
    int aryLength = sizeof(point) / sizeof(int);
    int *address = point;
    int sum = 0;
    for (int i = 0; i < aryLength; i++)
        sum += *(point + i);
    printf("메인에서 구한 합은 %d₩n", sum);
    printf("함수sumary() 호출로 구한 합은 %d₩n", sumary(point, aryLength));
    printf("함수sumary() 호출로 구한 합은 %d₩n", sumary(&point[0], aryLength
    printf("함수sumary() 호출로 구한 합은 %d₩n", sumary(address, aryLength))
    return 0;
}
```

```
//sum += *(point++);      //오류발생
//sum += *(address++);    //가능
```

[결과]
메인에서 구한 합은 755
함수sumary() 호출로 구한 합은 755
함수sumary() 호출로 구한 합은 755

다양한 배열원소 참조 방법 (arrayparam.c)

```
int sumary(int *ary, int SIZE)
{
    int sum = 0;
    for (int i = 0; i < SIZE; i++) {
        //sum += ary[i];
        //sum += *(ary + i);
        sum += *ary++;
        //sum += *(ary++);
    }
    return sum;
}
```

int sumary(int ary[], int SIZE)
도 가능

4가지가 모두 가능!

[결과]
메인에서 구한 합은 755
함수sumary() 호출로 구한 합은 755
함수sumary() 호출로 구한 합은 755
함수sumary() 호출로 구한 합은 755

배열 원소 수 계산 1/2
(arrayfunction.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
void readarray(double[], int);
void printarray(double[], int);
double sum(double[], int);
int main(void) {
    double data[5];
    int arraysize = sizeof(data) / sizeof(data[0]);
    printf("실수 %d개의 값을 입력하세요. ₩n", arraysize);
    readarray(data, arraysize);
    printf("₩n입력한 자료값은 다음과 같습니다.₩n");
    printarray(data, arraysize);
    printf("함수에서 구한 합은 %.3f 입니다.₩n", sum(data, arraysize));
    return 0;
}
```

[결과]

```
실수 5개의 값을 입력하세요.
data[0] = 4
data[1] = 3
data[2] = 7
data[3] = 8
data[4] = 9

입력한 자료값은 다음과 같습니다.
data[0] = 4.00 data[1] = 3.00 data
함수에서 구한 합은 31.000 입니다
```

배열 원소 수 계산 2/2
(arrayfunction.c)

```
void readarray(double data[], int n) {
    for (int i = 0; i < n; i++) {
        printf("data[%d] = ", i);
        scanf("%lf", &data[i]);
    }
}
void printarray(double data[], int n) {
    for (int i = 0; i < n; i++)
        printf("data[%d] = %.2lf ", i, *(data + i));
    printf("₩n");
}
double sum(double data[], int n) {
    double total = 0;
    for (int i = 0; i < n; i++)
        total += data[i];
    return total;
}
```

(data + i)
도 가능

*(data + i)
도 가능

2차원 배열을 함수 인자로 사용 1/2 (twodarrayfunction.c)

```
#include <stdio.h>

double sum(double data[][3], int, int);
void printarray(double data[][3], int, int);

int main(void) {
    double x[][3] = { { 1, 2, 3 },{ 7, 8, 9 },{ 4, 5, 6 },{ 10, 11, 12 } };

    int rowsize = sizeof(x) / sizeof(x[0]);
    int colsize = sizeof(x[0]) / sizeof(x[0][0]);
    printf("2차원 배열의 자료값은 다음과 같습니다.₩n");
    printarray(x, rowsize, colsize);
    printf("2차원 배열 원소합은 %.3lf 입니다.₩n", sum(x, rowsize, colsize));

    return 0;
}
```

[결과]

```
2차원 배열의 자료값은 다음과 같습니
 1행원소: x[0][0] =  1.00   x[0][1] =
 2행원소: x[1][0] =  7.00   x[1][1] =
 3행원소: x[2][0] =  4.00   x[2][1] =
 4행원소: x[3][0] = 10.00   x[3][1] =

2차원 배열 원소합은 78.000 입니다.
```

2차원 배열의 행/열 개수를 알아내어 함수에 전달.

2차원 배열을 함수 인자로 사용 1/2 (twodarrayfunction.c)

```
void printarray(double data[][3], int rowsize, int colsize) {
    for (int i = 0; i < rowsize; i++) {
        printf("% d행원소: ", i + 1);
        for (int j = 0; j < colsize; j++)
            printf("x[%d][%d] = %5.2lf    ", i, j, data[i][j]);
        printf("₩n");
    }
    printf("₩n");
}
double sum(double data[][3], int rowsize, int colsize) {
    double total = 0;
    for (int i = 0; i < rowsize; i++)
        for (int j = 0; j < colsize; j++)
            total += data[i][j];
    return total;
}
```