

C – 구조체(struct)

struct



구조체의 개념

◆ 구조체는 왜 필요할까?

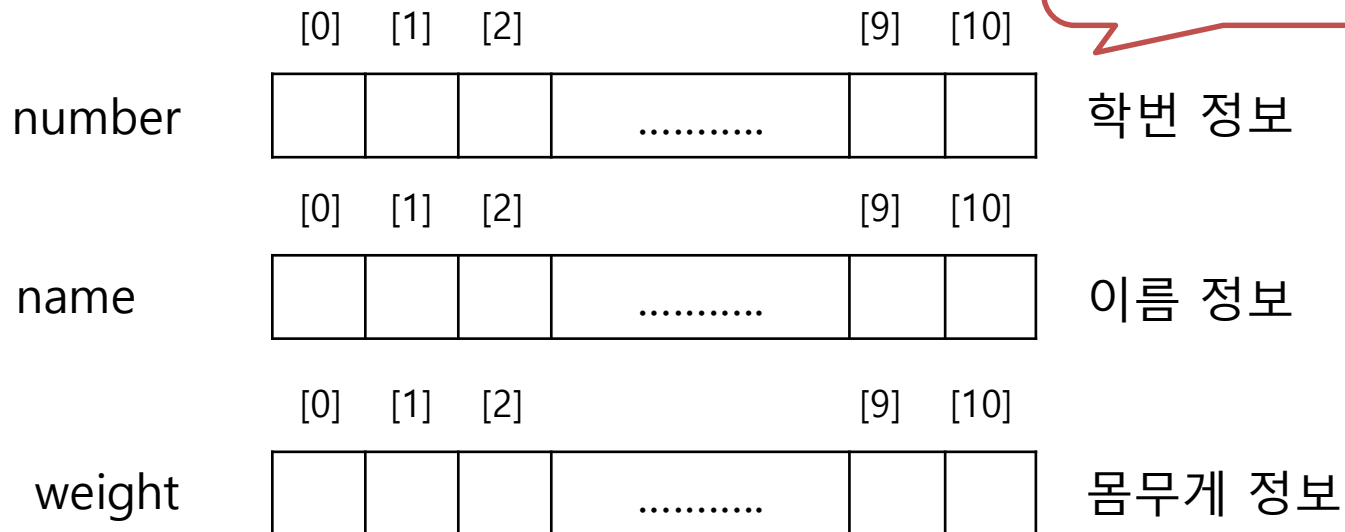
학생 10명의 학번과 이름, 몸무게 정보 저장 – 배열 자료형 이용

```
int number[10];
```

```
char name[20];
```

```
double weight[10];
```

정보가 흩어져서 저장되는 한계 발생!



구조체란 무엇인가?

◆ 구조체(structure)란?

다양한 자료형을 그룹화하여 하나의 변수로 처리할 수 있게 만든 자료형이다.
개발자가 다양한 정보를 저장하기 위해 필요에 따라 생성하는 자료형을 **사용자 정의 자료형** 또는 **구조체**라 한다.

- 구조체 정의

```
struct 구조체이름{  
    자료형 멤버이름;  
};
```

- 구조체 변수 선언

```
struct 구조체이름 변수이름;
```



구조체의 정의 및 사용

◆ 구조체 정의

```
struct Person {  
    //이름, 나이, 키  
    char name[20];  
    int age;  
    float height;  
};
```

※ 멤버 변수는 일반 변수
처럼 초기화 할 수 없음
int age = 0 (x)

- 구조체 객체(변수) 선언

```
struct Person p1
```

- 구조체 객체(변수) 선언과 초기화

```
struct Person p1 = {"알파고", 11, 170.4f}
```



구조체의 정의 및 사용

◆ 구조체 변수 생성 및 사용

```
//구조체 변수 선언
struct Person p1;

//p1.name = "알파고";
strcpy(p1.name, "알파고");
p1.age = 11;
p1.height = 170.5f;

//구조체 변수 선언과 초기화
//struct Person p1 = { "알파고", 11, 165.5f };

printf("이름: %s\n", p1.name);
printf("나이: %d\n", p1.age);
printf("키: %.1f\n", p1.height);
```



구조체 배열

◆ 구조체 배열 – 객체를 여러 개 생성

```
struct Person p[3] = {  
    {"이산", 15, 171.9f},  
    {"한강", 35, 163.3f},  
    {"박봄", 22, 178.4f},  
};  
int i;
```

구조체 배열 선언과
동시에 초기화

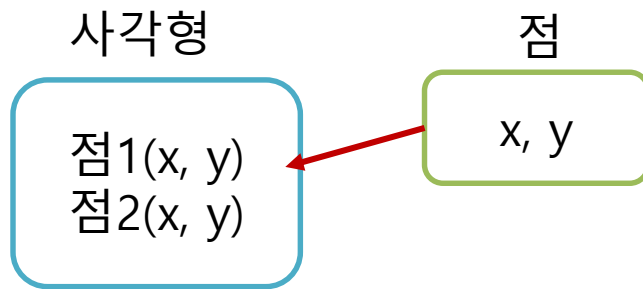
```
//p[0]의 정보  
/*printf("이름: %s\n", p[0].name);  
printf("나이: %d\n", p[0].age);  
printf("키: %.1f\n", p[0].height);*/  
  
for (i = 0; i < 3; i++)  
{  
    printf("이름: %s, 나이: %d, 키: %.1f\n",  
        p[i].name, p[i].age, p[i].height);  
}
```



중첩 구조체

◆ 중첩 구조체

구조체의 멤버 변수가 다른 구조체 자료형인 경우



```
//점(좌표) 구조체 정의
struct Point
{
    int x;
    int y;
};

//사각형 구조체 정의
struct Rectangle
{
    //Point 구조체 자료형 참조
    struct Point p1;
    struct Point p2;
};
```



중첩 구조체

◆ 중첩 구조체

```
int main()
{
    //두 점을 이용한 직사각형 만들기
    //좌측 상단 좌표, 우측 하단 좌표 생성

    /*struct Rectangle rect;
    rect.p1.x = 1;
    rect.p1.y = 5;
    rect.p2.x = 4;
    rect.p2.y = 2;*/

    struct Rectangle rect = {
        .p1 = {1, 5},
        .p2 = {5, 2}
    };
    int width, height;
```

점 1(1, 5), 점 2(5, 2)
너비:4, 높이: 3



중첩 구조체

◆ 중첩 구조체

```
//좌표 출력
printf("점1(%d, %d), 점2(%d, %d)\n", rect.p1.x, rect.p1.y,
      rect.p2.x, rect.p2.y);

//너비와 높이 계산
width = abs(rect.p2.x - rect.p1.x);
height = abs(rect.p2.y - rect.p1.y);

printf("너비:%d, 높이: %d\n", width, height);

return 0;
}
```



구조체 typedef 키워드 사용

- **typedef struct** 구조체

typedef 를 이용한 구조체 정의하면 구조체 변수 선언시 struct를 생략할 수 있어 코드가 간결해 짐

```
typedef struct {  
    자료형 멤버이름;  
} 구조체이름;
```



```
typedef struct {  
    char name[20],  
    int age;  
    float height;  
} Person;
```

```
Person p1;
```



구조체 typedef 키워드 사용

- typedef struct 구조체

```
#define _CRT_SECURE_NO_WARNINGS //strcpy()
#include <stdio.h>

//Employee 구조체 정의
typedef struct {
    int id;           //사원 아이디
    char name[20];    //사원 이름
    int salary;       //급여
}Employee;
```



구조체 typedef 키워드 사용

- typedef struct 구조체

```
//구조체 변수 선언
Employee e1;

//입력(초기화)
e1.id = 1;
strcpy(e1.name, "이사원");
e1.salary = 3000000;

//Employee e1 = { 1, "이사원", 3000000 };

//정보 출력
printf("사원 ID: %d, 이름: %s, 급여: %d\n",
      e1.id, e1.name, e1.salary);
```



구조체 typedef 키워드 사용

- **typedef struct** 구조체 – 함수 사용

```
//사원 구조체 정의
typedef struct {
    int id;
    char name[20];
    int salary;
}Employee;

//사원 정보 출력
void printInfo(Employee e) {
    printf("사원 ID: %d, 이름: %s, 급여: %d\n",
        e.id, e.name, e.salary);
}
```



구조체 typedef 키워드 사용

- typedef struct 구조체 – 함수 사용

```
int main()
{
    //구조체 변수 선언
    Employee emp1;

    //입력
    emp1.id = 1001;
    strcpy(emp1.name, "박상희");
    emp1.salary = 3000000;

    //출력
    printInfo(emp1);
}
```



구조체 typedef 키워드 사용

- typedef struct 구조체 배열

```
//구조체 배열 선언
Employee employees[3] = {
    {1001, "임시연", 2500000},
    {1002, "우상영", 2000000},
    {1003, "이정우", 3000000}
};

printf("===== 사원 명단 =====\n");
for (int i = 0; i < 3; i++) {
    printInfo(employees[i]);
}

return 0;
}
```

```
사원 ID: 1001, 이름: 박상희, 급여: 3000000
===== 사원 명단 =====
사원 ID: 1001, 이름: 임시연, 급여: 2500000
사원 ID: 1002, 이름: 우상영, 급여: 2000000
사원 ID: 1003, 이름: 이정우, 급여: 3000000
```



은행 업무 프로젝트 개요

◆ 은행 업무 프로젝트

은행 계좌 구조체를 만들고, 은행 업무 기능 만들기

- 은행 업무 프로젝트 단계

step1. 문제 정의하기

step2. 구조체 정의하고 관계도 그리기

step3. 은행 업무 기능 설계하고 구현하기

step4. 파일 배포(서비스)



step1. 문제 정의하기

◆ 프로그램 시나리오

- 계정(BankAccount) 구조체는 계좌 번호, 계좌주, 잔액으로 구성되어 있다.
- BankAccount 배열을 100개 생성한다.
- main 함수에는 계좌 생성, 입금, 출금, 계좌 목록, 종료 등의 메뉴가 있다.

계좌 번호	계좌주	금액
100-200-3000	한강	1000
101-200-3000	이이슬	2000
102-200-3000	알파고	3000



step1. 문제 정의하기

◆ 메뉴별 결과 화면

```
=====
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====
선택 > 1
계좌 번호 : 100-200-3000
계좌주 : 한이슬
결과 : 계좌가 생성되었습니다. (계좌 번호 : 100-200-3000)
```

```
=====
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====
선택 > 2
입금할 계좌번호(예 : 000-000-0000) : 100-200-3000
입금액 : 20000
정상 처리 되었습니다. 현재 잔액 : 20000
```

```
=====
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====
선택 > 3
출금할 계좌번호(예 : 000-000-0000) : 100-200-3000
출금액 : 10000
정상 처리 되었습니다. 현재 잔액 : 10000
```



step1. 문제 정의하기

◆ 메뉴별 결과 화면

```
=====
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====
선택> 4
***** 계좌 목록 *****
계좌 번호 : 100-200-3000, 계좌주 : 한이슬, 잔액 : 10000
```

@ 출금시 잔액 부족 오류

```
=====
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====
선택> 3
출금할 계좌번호(예 : 000-000-0000) : 100-200-3000
출금액 : 50000
잔액이 부족하거나 올바른 금액이 아닙니다. 현재 잔액 : 20000
```

@ 계좌 중복 오류

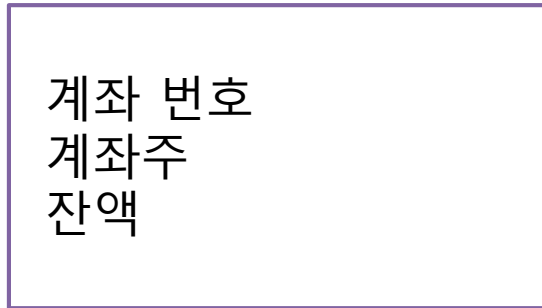
```
=====
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====
선택> 1
계좌 번호 : 100-200-3000
이미 등록된 계좌입니다. 다시 입력하세요.
```



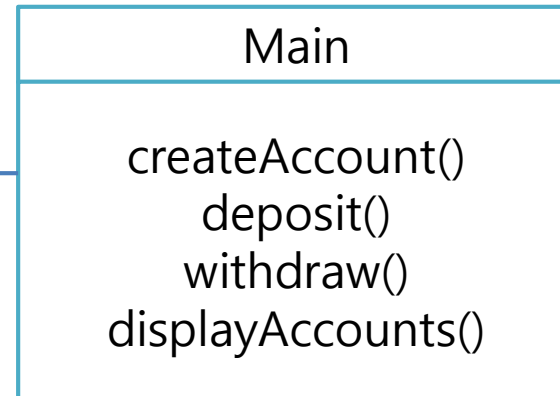
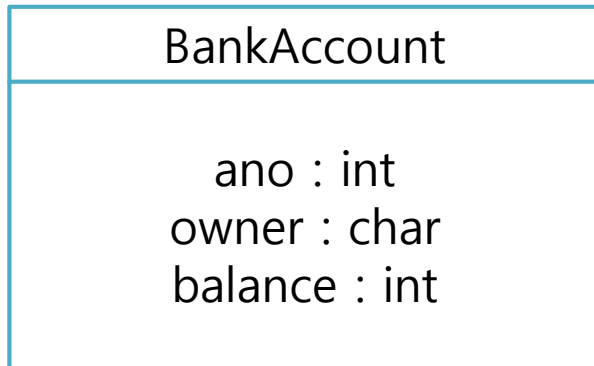
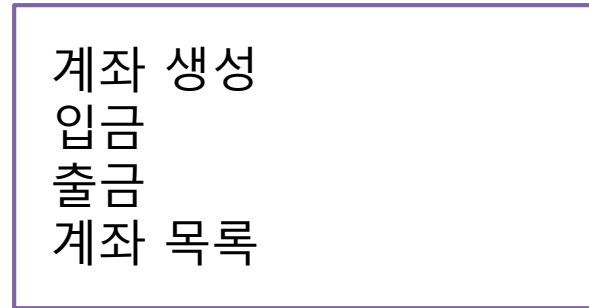
step2. 구조체 다이어그램

◆ 구조체 관계도 그리기

BankAccount 구조체



main 함수



step2. 구조체 정의하기

▪ BankAccount 구조체

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h> //strcpy()
#include <stdbool.h> //true/false 사용

#define MAX_ACCOUNTS 100 //최대 계정수
#define ANO_LEN 20 //계좌번호 크기
#define OWNER_LEN 30 //예금주 크기

typedef struct {
    char ano[ANO_LEN]; //계좌번호
    char owner[OWNER_LEN]; //예금주
    int balance; //잔고
}BankAccount;

//전역 공간
BankAccount accounts[MAX_ACCOUNTS]; //계좌 배열 생성
int accountCount = 0; //현재 계좌 수
```



step3. 은행 업무 기능 설계, 구현

■ main() 함수

```
int main()
{
    bool run = true; //실행,종료
    int choice; //메뉴

    while (run)
    {
        printf("=====\n");
        printf("1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료\n");
        printf("=====\n");
        printf("선택> ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                createAccount();
                break;
            case 2:
                deposit();
                break;
```



step3. 은행 업무 기능 설계, 구현

▪ main() 함수

```
        case 3:
            withdraw();
            break;
        case 4:
            displayAccounts();
            break;
        case 5:
            printf("프로그램을 종료합니다.\n");
            run = false;
            break; //정상 종료
        default:
            printf("잘못된 입력입니다. 다시 선택하세요.\n");
    }
}

system("pause"); //콘솔창 닫힘 방지(파일 배포시)

return 0;
}
```



step3. 은행 업무 기능 설계, 구현

■ 계좌 생성

```
void createAccount() {
    char accountNumber[ANO_LEN]; //입력(계좌번호)

    if (accountCount >= MAX_ACCOUNTS) {
        printf("더 이상 계좌를 생성할 수 없습니다.\n");
        return;
    }

    printf("계좌 번호(예: xx-xx-xxxx): ");
    scanf("%s", accountNumber);

    //중복 검사
    for (int i = 0; i < accountCount; i++) {
        if (strcmp(accounts[i].ano, accountNumber) == 0) {
            printf("이미 등록된 계좌입니다. 다시 입력하세요.\n");
            return;
        }
    }
}
```



step3. 은행 업무 기능 설계, 구현

■ 계좌 생성

```
//중복 없을때 계좌 생성
strcpy(accounts[accountCount].ano, accountNumber); //계좌번호 복사
printf("계좌주: ");
scanf("%s", accounts[accountCount].owner);
accounts[accountCount].balance = 0; //잔고

printf("결과: 계좌가 생성되었습니다. (계좌 번호: %s)\n",
       accounts[accountCount].ano);

accountCount++; //다음 인덱스로 증가
}
```



step3. 은행 업무 기능 설계, 구현

■ 계좌 목록

```
void displayAccounts() {  
    printf("***** 계좌 목록 *****\n");  
  
    if (accountCount == 0)  
    {  
        printf("등록된 계좌가 없습니다.\n");  
        return;  
    }  
  
    for (int i = 0; i < accountCount; i++)  
    {  
        printf("계좌 번호: %s, 계좌주: %s, 잔액: %d\n",  
            accounts[i].ano, accounts[i].owner, accounts[i].balance);  
    }  
}
```



step3. 은행 업무 기능 설계, 구현

■ 예금

```
void deposit() {
    char accountNumber[ANO_LEN]; //외부 입력(계좌번호)
    int amount; //입금액

    printf("입금할 계좌번호(예: xx-xx-xxxx): ");
    scanf("%s", accountNumber);

    for (int i = 0; i < accountCount; i++) {
        //이미 등록된 계좌와 입력 계좌가 일치하면
        if (strcmp(accounts[i].ano, accountNumber) == 0) {
            printf("입금액: ");
            scanf("%d", &amount);
            if (amount < 0) {
                printf("올바른 금액이 아닙니다.\n");
                return; //즉시 종료
            }
        }
    }
}
```



step3. 은행 업무 기능 설계, 구현

▪ 예금

```
    else {  
        accounts[i].balance += amount;  
        printf("정상 처리 되었습니다. 현재 잔액: %d\n",  
               accounts[i].balance);  
        return; //for문 탈출  
    }  
}  
}  
printf("계좌를 찾을 수 없습니다.\n");  
}
```



step3. 은행 업무 기능 설계, 구현

■ 출금

```
void withdraw() {
    char accountNumber[ANO_LEN]; //외부 입력(계좌번호)
    int amount; //출금액

    printf("출금할 계좌번호(예: xx-xx-xxxx): ");
    scanf("%s", accountNumber);

    for (int i = 0; i < accountCount; i++) {
        if (strcmp(accounts[i].ano, accountNumber) == 0) {
            printf("출금액: ");
            scanf("%d", &amount);
            if (amount > accounts[i].balance || amount < 0) {
                printf("잔액이 부족하거나 올바른 금액이 아닙니다. "
                    "현재 잔액: %d\n", accounts[i].balance);
                return;
            }
        }
    }
}
```



step3. 은행 업무 기능 설계, 구현

■ 출금

```
        else {
            accounts[i].balance -= amount;
            printf("정상 처리 되었습니다. 현재 잔액: %d\n",
                  accounts[i].balance);
            return; //for문 탈출
        }
    }
}
printf("계좌를 찾을 수 없습니다.\n");
}
```



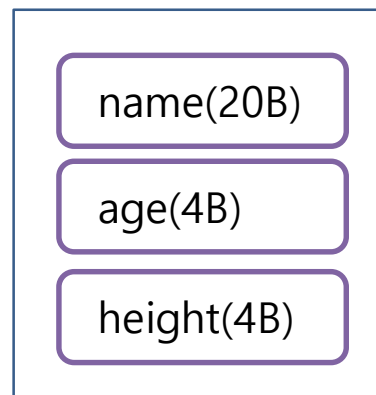
구조체 포인터 변수

- 구조체 포인터 사용

구조체 변수가 닷(.) 연산자를 사용하는 반면, 구조체의 포인터 변수는 참조 연산자(->)를 사용한다.

```
typedef struct
{
    //이름, 나이, 키
    char name[20];
    int age;
    float height;
}Person;
```

Person



ptr(8B)

Person의 주소저장

```
Person p1 = { "알파고", 11, 171.9f };
Person* ptr = &p1; //구조체 포인터 선언
```

ptr->name, ptr->age, ptr->height



구조체 typedef 키워드 사용

- 구조체 포인터 사용

```
typedef struct {  
    int no;  
    char name[20];  
    int age;  
}Hero;  
  
int main()  
{  
    //구조체 변수 선언과 초기화  
    Hero p1 = { 1, "고담덕", 39 };  
  
    //점(.) 연산자로 속성에 접근함  
    printf("번호: %d, 이름: %s, 나이: %d\n",  
           p1.no, p1.name, p1.age);  
}
```



구조체 typedef 키워드 사용

- 구조체 포인터 사용

```
//구조체 포인터 선언과 초기화
Hero p2 = { 2, "이순신", 54 };
Hero* ptr = &p2;

//화살표(->) 연산자로 속성에 접근함
printf("번호: %d, 이름: %s, 나이: %d\n",
      ptr->no, ptr->name, ptr->age);

return 0;
}
```

```
번호 : 1, 이름 : 고담덕, 나이 : 39
번호 : 2, 이름 : 이순신, 나이 : 54
```



날짜와 시간 구현 구조체

- 현재 날짜와 시간 표시하기

```
#define _CRT_SECURE_NO_WARNINGS //localtime()
#include <stdio.h>
#include <time.h>

int main()
{
    time_t ct = time(NULL); //현재 시간
    struct tm* now = localtime(&ct); //현재 날짜와 시간 저장

    printf("현재 날짜: %d. %d. %d.\n", now->tm_year + 1900,
        now->tm_mon + 1, now->tm_mday);

    printf("현재 시간: %d : %d : %d.\n", now->tm_hour,
        now->tm_min, now->tm_sec);
}
```

```
현재 날짜: 2025. 5. 20.
현재 시간: 11 : 19 : 26.
오전 11시 19분 26초
현재 요일: 2
화요일입니다.
```



날짜와 시간 구현 구조체

- 현재 날짜와 시간 표시하기

```
//12시각제 사용
int hour = (now->tm_hour > 12) ? now->tm_hour - 12 : now->tm_hour;
char* ampm = (now->tm_hour < 12) ? "오전" : "오후";
printf("%s %d시 %d분 %d초\n", ampm, hour, now->tm_min, now->tm_sec);

//요일 (0-일, 1-월, 2-화...)
printf("현재 요일: %d\n", now->tm_wday);

//요일 출력
char* days[] = {"일", "월", "화", "수", "목", "금", "토"};
int idx = now->tm_wday;

printf("%s요일입니다.\n", days[idx]);

return 0;
}
```



구조체 배열과 포인터 변수

● 구조체 배열과 포인터 변수

```
//과일 구조체
typedef struct
{
    char name[20];
    int quantity;
    char* type;
}Fruit;
```

```
===== 변수로 접근 =====
과일 이름: 대구 사과
수량: 100
과일 종류: Grape
===== 포인터로 접근 =====
과일 이름: 대구 사과
수량: 100
과일 종류: Grape
```

```
//포인터 배열 선언
char* types[] = { "Apple", "Banana", "Orange" };

Fruit f = { "대구 사과", 100, types[1]}; //변수 할당
Fruit* ptr; //구조체 포인터 할당

printf("===== 변수로 접근 =====\n");
printf("과일 이름: %s\n", f.name);
printf("수량: %d\n", f.quantity);
f.type = "Grape"; //과일 종류 변경
printf("과일 종류: %s\n", f.type);

ptr = &f; //f의 주소 저장

printf("===== 포인터로 접근 =====\n");
printf("과일 이름: %s\n", ptr->name);
printf("수량: %d\n", ptr->quantity);
f.type = "Grape"; //과일 종류 변경
printf("과일 종류: %s\n", ptr->type);
```

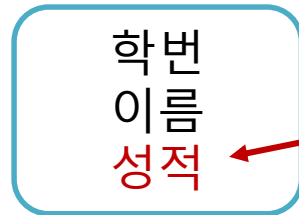


성적 관리 프로그램

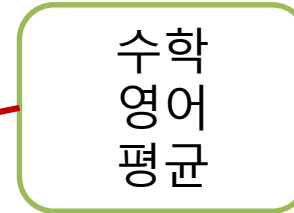
◆ 성적 관리 프로그램

- 학생 구조체가 성적 구조체 자료형을 사용함

학생



성적



```
typedef struct
{
    int number;
    char name[20];
    Score score;
}Student;
```

```
typedef struct
{
    int math;
    int eng;
    double avg;
}Score;
```



성적 관리 프로그램

```
typedef struct {  
    int math;  
    int eng;  
    double avg;  
}Score;  
  
typedef struct {  
    int number;  
    char name[20];  
    Score score; //Score 구조체 포함  
}Student;  
  
void calcAvg(Score* score) { //포인터 - 참조에 의한 전달  
    score->avg = (score->math + score->eng) / 2.0;  
}  
  
void showStudentInfo(Student* st) { //포인터 - 복사시 용량 축소  
    printf("학번: %d, 이름: %s\n", st->number, st->name);  
    printf("수학: %d, 영어: %d\n", st->score.math, st->score.eng);  
    printf("평균: %.11f\n", st->score.avg);  
}
```



성적 관리 프로그램

- 구조체 변수 선언

```
int main()
{
    //구조체 변수 선언과 초기화
    Student s1 = {
        .number = 101,
        .name = "임시현",
        .score = {95, 88, 0}
    };

    //성적의 평균
    calcAvg(&s1.score);

    //학생의 정보
    showStudentInfo(&s1);

    return 0;
}
```

학번 : 101, 이름 : 임시현
수학 : 95, 영어 : 88
평균 : 91.5



성적 관리 프로그램

- 구조체 배열 선언

```
int main()
{
    //구조체 배열 선언과 초기화
    Student students[3] = {
        {101, "임시현", {95, 88, 0.0}},
        {102, "이정후", {80, 95, 0.0}},
        {103, "신유빈", {85, 90, 0.0}},
    };

    //성적의 평균과 정보
    for (int i = 0; i < 3; i++) {
        calcAvg(&students[i].score);
        showStudentInfo(&students[i]);
    }

    return 0;
}
```

```
학번 : 101, 이름 : 임시현
수학 : 95, 영어 : 88
평균 : 91.5
학번 : 102, 이름 : 이정후
수학 : 80, 영어 : 95
평균 : 87.5
학번 : 103, 이름 : 신유빈
수학 : 85, 영어 : 90
평균 : 87.5
```

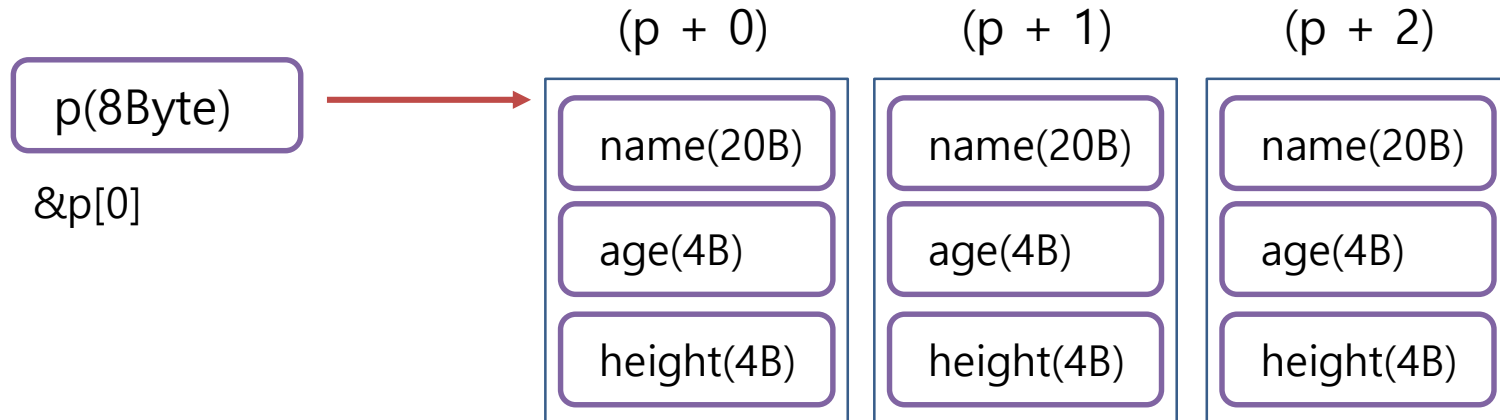


구조체 배열 동적 할당

- 구조체 포인터 사용 필요성

1. 함수의 매개 변수로 구조체 복사(전달)시 용량과 시간을 줄일 수 있음 (모든 포인터의 크기는 8byte 이다)
2. 동적 메모리 할당이 가능하다.

```
Person* p = (Person*)malloc(sizeof(Person) * 3);
```



구조체 배열 동적 할당

- 구조체 배열 동적 할당

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

//Employee 구조체 정의

```
typedef struct {
    int id;
    char name[20];
    int salary;
}Employee;
```

//사원 정보 출력 함수

```
void displayInfo(Employee* e, int len) {
    for (int i = 0; i < len; i++) {
        printf("사원 ID: %d, 이름: %s, 급여: %d\n",
            (e + i)->id, (e + i)->name, (e + i)->salary);
    }
}
```

```
사원 ID: 1001, 이름: 강사원, 급여: 2000000
사원 ID: 1002, 이름: 박대리, 급여: 3000000
사원 ID: 1003, 이름: 한과장, 급여: 4000000
```



구조체 배열 동적 할당

- 구조체 배열 동적 할당

```
int main()
{
    //구조체 배열 동적 할당
    Employee* emp;
    emp = (Employee *)malloc(sizeof(Employee) * 3);
    if (emp == NULL) {
        printf("동적 메모리 할당에 실패했습니다.\n");
        exit(1);
    }

    //emp 1명 생성
    emp->id = 1001;
    strcpy_s(emp->name, sizeof(emp->name), "강사원");
    emp->salary = 2000000;
}
```



구조체 배열 동적 할당

- 구조체 배열 동적 할당

```
//emp 2명 생성
(emp + 1)->id = 1002,
strcpy_s((emp + 1)->name, sizeof(emp->name), "박대리");
(emp + 1)->salary = 3000000;

//emp 3명 생성
(emp + 2)->id = 1003,
strcpy_s((emp + 2)->name, sizeof(emp->name), "한과장");
(emp + 2)->salary = 4000000;

displayInfo(emp, 3); //displayInfo() 호출

free(emp); //메모리 해제

return 0;
}
```



구조체 배열 동적 할당

- 구조체 배열 동적 할당 – 배열 인덱스로 접근

```
//사원 정보 출력 함수
void displayInfo(Employee* e, int len) {
    for (int i = 0; i < len; i++) {
        printf("사원 ID: %d, 이름: %s, 급여: %d\n",
            e[i].id, e[i].name, e[i].salary);
    }
}

int main()
{
    //구조체 배열 동적 할당
    Employee* emp;
    emp = (Employee*)malloc(sizeof(Employee) * 3);
    if (emp == NULL) {
        printf("동적 메모리 할당에 실패했습니다.\n");
        exit(1);
    }
}
```



구조체 배열 동적 할당

- 구조체 배열 동적 할당 – 배열 인덱스로 접근

```
//emp 1명 생성
emp[0].id = 1001;
strcpy_s(emp[0].name, sizeof(emp[0].name), "강사원");
emp[0].salary = 2000000;

//emp 2명 생성
emp[1].id = 1002;
strcpy_s(emp[1].name, sizeof(emp[1].name), "박대리");
emp[1].salary = 3000000;

//emp 3명 생성
emp[2].id = 1003;
strcpy_s(emp[2].name, sizeof(emp[2].name), "한과장");
emp[2].salary = 4000000;


displayInfo(emp, 3); //displayInfo() 호출

free(emp); //메모리 해제
return 0;
}
```



실습 문제 – 구조체 배열 동적 할당

Book 구조체를 만들어서 책 2권을 출력하는 프로그램을 작성하세요

실행결과 

```
번호 : 201, 제목 : 모두의 C언어  
번호 : 202, 제목 : 채식주의자
```

