

11. 전처리문과 구조체활용

Autumn 2019

Today

1. 지난 시간 복습 - 전처리문과 구조체

2. 메모리 할당하기

3. 실습

전처리문 (preprocessor)

- 컴파일하기 전에 미리 처리되는 문장
- 소스코드 시작부분에 위치하며, #으로 시작
- 일반적으로 미리 정의되어 있음을 알아보기 쉽도록 대문자를 사용함
- 종류:
#include, #define, #ifdef, #ifndef, #endif 등 다수

예제. #include <stdio.h>

#define

- 소스코드에서 사용할 숫자, 문자열, 함수이름이 너무 길고 복잡할 때 상수처럼 사용할 기호를 정의해서 사용
- 사용법: #define [기호] [숫자, 문자열, 함수]

- 예시1.

```
#include <stdio.h>

int main(){

    double r = 3.0;
    double area = r * r * 3.1415926535;

}
```

```
#include <stdio.h>
#define PI 3.1415926535

int main(){

    double r = 3.0;
    double area = r * r * PI;

}
```

- 예시2.

```
#include <stdio.h>
#define PI 3.1415926535
#define STR "원의 면적을 계산했습니다."
#define END_MSG printf("함수를 종료합니다.")

int main(){

    double r = 3.0;
    double area = r * r * PI;

    puts(STR);
    END_MSG;

}
```

숫자 전처리문

문자열 전처리문

함수 전처리문

#ifdef #ifndef #endif

- 문법의 조건문과 비슷하게 작동함
 - 이미 선언된 기호가 있는지 체크해서 컴파일 여부를 결정함
- 조건문 전처리로, 컴파일 여부를 결정하도록 함

```
#ifdef BLOCK
#include <stdio.h>
#define PI (3.1415926535)
#define STR ("원의 면적을 계산했습니다.")
#define END_MSG (printf("함수를 종료합니다. "))

int main() {

    double r = 3.0;
    double area = r * r * PI;

    puts(STR);
    END_MSG;
}
#endif
```

```
#define BLOCK
#ifdef BLOCK
#include <stdio.h>
#define PI (3.1415926535)
#define STR ("원의 면적을 계산했습니다.")
#define END_MSG (printf("함수를 종료합니다. "))

int main() {

    double r = 3.0;
    double area = r * r * PI;

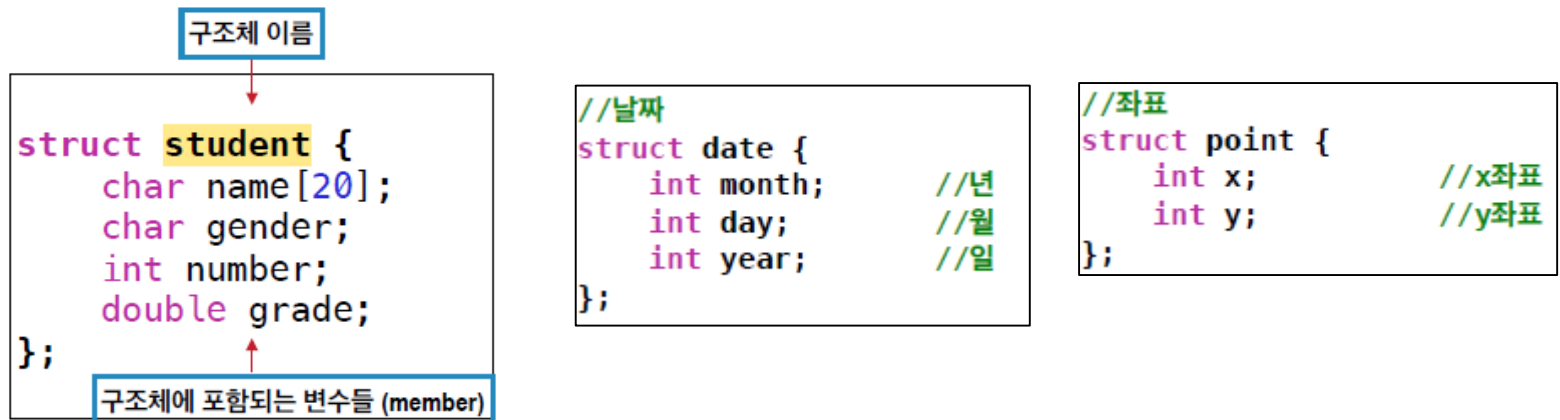
    puts(STR);
    END_MSG;
}
#endif
```

구조체 (struct)

- 서로 다른 변수들을 하나로 묶어주는 기능

이름: 김xx 성별: 남 학번: 20170111 학과: 산업공학 학점: 4.50 ..	이름: 이xx 성별: 여 학번: 20131234 학과: 행정학과 학점: 3.23 ..	이름: 박xx 성별: 여 학번: 20151234 학과: 경영학과 학점: 2.37 ..	이름: 최xx 성별: 남 학번: 20161234 학과: 철학과 학점: 3.85 ..	이름: char [] 성별: char (M/F) 학번: int 학과: int (0,1,...) 학점: double
--	--	--	---	--	------

- 혹은 비슷한 기능을 하는 변수끼리 묶어주고 싶을 때,



구조체 배열

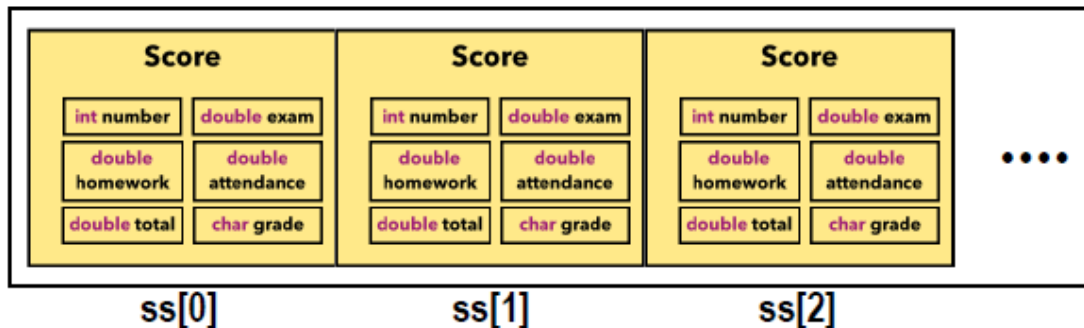
- ▶ 앞의 예제에서 학생 1명이 아니라 반 전체 40명의 점수 데이터를 저장하여 관리하려고 한다. 이때 40명에 대한 Score 구조체 변수를 각각 선언하는 것은 비효율적이다. 100명, 1000명이 된다면?

구조체를 배열로 선언할 수 있다

```
Score ss[40];
```

Score 구조체 40개를 배열로 선언

SS



구조체를 사용하는 이유는 기본적으로 많은 양의 데이터를 효율적으로 저장하고 관리하기 위해서이기 때문에, 구조체는 배열의 형태로 활용하는 것이 일반적임

구조체 포인터

- 구조체 포인터 변수는 구조체의 주소값을 저장한다.

❶ 일반 포인터 변수 사용

```
int a;  
int* p;  
p = &a;  
*p = 100;
```

❷ 구조체 포인터 변수 사용

```
struct student {  
    char name[10];  
    int kor;  
    int eng;  
    float avg;  
};  
struct student s;  
struct student *p;  
p = &s;  
p->kor = 100;
```


Today

1. 지난 시간 복습 – 전처리문과 구조체

2. 메모리 할당하기

3. 실습

다양한 구조체 멤버 (member)

- 구조체는 서로 다른 데이터형을 요소로 가짐

1. 구조체도 요소로 가질 수 있음

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

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

```
struct triangle {  
    struct point ptA;  
    struct point ptB;  
    struct point ptC;  
};
```

← 구조체의 변수로 구조체를 가짐

```
int main() {
```

```
    struct triangle t = { {1,2},{-1,2},{0,4} };
```

```
    printf("point A: (%d,%d) \t point B: (%d,%d) \t point C: (%d,%d) \n",  
        t.ptA.x, t.ptA.y, t.ptB.x, t.ptB.y, t.ptC.x, t.ptC.y);
```

→ 괄호 없이 순서대로 써도 초기화가 가능

```
    struct triangle t = { 1,2,-1,2,0,4 };
```

```
}
```

2. 구조체포인터도 요소로 가질 수 있음

```
struct chain_e {  
    int data;  
    struct chain_element* next;  
};
```

구조체 배열

```
#include <stdio.h>

struct point {
    int x;
    int y;
};

struct triangle {
    struct point ptA;
    struct point ptB;
    struct point ptC;
};

int main() {

    struct triangle t = { {1,2},{-1,2},{0,4} };
    printf("point A: (%d,%d) \t point B: (%d,%d) \n",
        t.ptA.x, t.ptA.y, t.ptB.x, t.ptB.y);
}
```



```
#include <stdio.h>

struct point {
    int x;
    int y;
};

struct triangle {
    struct point pts[3];
};

int main() {

    int i = 0;
    struct triangle t = { {1,2},{-1,2},{0,4} };

    for (i = 0; i < 3; i++) {
        printf("point %d: (%d,%d) \n", i, t.pts[i].x, t.pts[i].y);
    }
}
```

구조체 포인터

- 구조체의 변수가 많은 경우에는 구조체포인터를 쓰는 것이 효율적임
- 구조체포인터로 멤버변수에 접근할 때는 ➔ 를 사용함

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

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

```
void func(struct point *p) {  
    printf("x: %d, y: %d", p->x, p->y);  
}
```

```
int main() {  
    struct point p = { 1, 2 };  
    func(&p);  
}
```

← 구조체포인터를 함수의 매개변수로 사용

← 구조체포인터로 변수 접근 시 ➔ 사용

변수 접근은 다음과 같이 사용할 수도 있음

```
printf("x: %d, y: %d", (*p).x, (*p).y);
```

Today

1. 지난 시간 복습 – 전처리문과 구조체
2. 메모리 할당하기
3. 실습

[실습1] 구조체

- 수강신청 프로그램에 교과목정보를 입력하는 구조체를 만들기
- 과목명은 길이 20, 교수이름은 길이 10으로 정의하시오.
- 교과목정보 구조체를 만들고, 아래 데이터 하나를 저장하고, 추가로 신청 가능한 학생수를 출력하시오.

과목명: Programming1

교수: Han

학년: 1

학점: 3

정원: 40

신청인원: 40

Programming1 과목의 여석은 0입니다.

[실습2] 구조체 배열

- 앞의 실습에서 과목을 여러 개 받을 수 있도록 구조체 배열로 선언해보시오
- 다음의 정보를 키보드로 입력받고, 결과와 같이 출력하시오

과목명: Programming1

교수: Han

학년: 1

학점: 3

정원: 40

신청인원: 40

과목명: Calculus

교수: Kim

학년: 1

학점: 3

정원: 55

신청인원: 45

과목명: Chemistry

교수: Lee

학년: 1

학점: 3

정원: 27

신청인원: 25

과목명: English

교수: Park

학년: 1

학점: 2

정원: 20

신청인원: 17

```
1번째 교과목 정보 입력: 과목명, 교수, 학년, 학점, 정원, 신청인원
Programming1 Han 1 3 40 40
2번째 교과목 정보 입력: 과목명, 교수, 학년, 학점, 정원, 신청인원
Calculus Kim 1 3 55 45
3번째 교과목 정보 입력: 과목명, 교수, 학년, 학점, 정원, 신청인원
Chemistry Lee 1 3 27 25
4번째 교과목 정보 입력: 과목명, 교수, 학년, 학점, 정원, 신청인원
English Park 1 2 20 17
```

[실습3] 구조체 함수

- 구조체를 **매개변수로** 사용해서 구조체의 정보를 출력하는 함수를 작성하시오

```
void printStatus(Course c)
{
    ..
}
```

- main() 에서 위의 함수를 호출해서 다음과 같이 결과를 출력해보시오 (printf 함수를 main 에서 사용하지 않습니다)

[수강신청 현황]

Programming1 - Han - 수강신청마감

Calculus - Kim - 10명 신청가능

Chemistry - Lee - 2명 신청가능

English - Park - 3명 신청가능

[실습4] 구조체포인터 함수

- 구조체포인터를 **매개변수로** 사용해서 구조체의 정보를 출력하는 함수를 작성하시오

```
void printStatus(Course c)
{
    ..
}
```

- main() 에서 위의 함수를 호출해서 다음과 같이 결과를 출력해보시오 (printf 함수를 main 에서 사용하지 않습니다)

[수강신청 현황]

Programming1 - Han - 수강신청마감

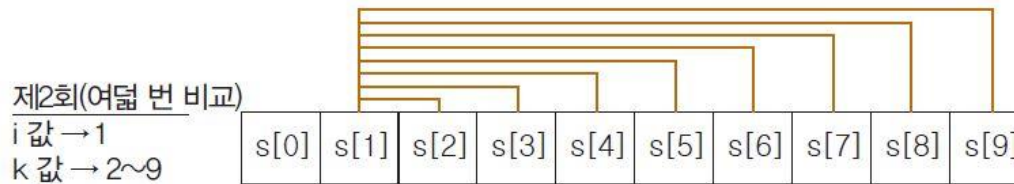
Calculus - Kim - 10명 신청가능

Chemistry - Lee - 2명 신청가능

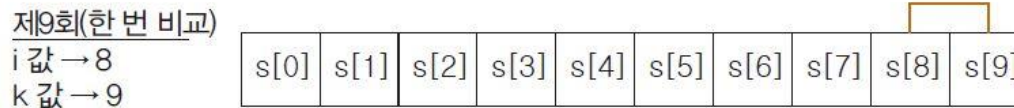
English - Park - 3명 신청가능

[실습26] 포인터를 이용한 배열 정렬

예제 설명 포인터를 이용하여 배열에 들어 있는 값 10개를 정렬하는 프로그램이다. 다음 그림을 참고하여 두 값을 비교하고 작은 것을 앞으로 옮기는 선택 정렬을 사용한다.



⋮



실행 결과

```
C:\Windows\system32\cmd.exe
정렬 전 배열 s ==> 1 0 3 2 5 4 7 6 9 8
정렬 후 배열 s ==> 0 1 2 3 4 5 6 7 8 9
```

[실습1]

- ▶ **int**형 배열에 **0**에서 **999**까지 **random number**를 **10**개 생성하여 저장한 후, 최소값을 계산하는 프로그램을 작성해 보세요.

```
0: 807
1: 249
2: 73
3: 658
4: 930
5: 272
6: 544
7: 878
8: 923
9: 709
최소값은 73입니다.
```

[실행 예]

[실습2]

- ▶ **int**형 배열에 0에서 999까지 **random number**를 100개 생성하여 저장한 후, 100단위별로 생성된 숫자들의 빈도수를 출력해 보세요.
- ▶ **Hint.** 앞의 예제를 확장하여 빈도수를 저장할 크기가 10인 **int**형 배열 **bb**를 생성하여 모두 0으로 초기화한다. 다음으로 **for**문을 사용하여 100개의 숫자들을 하나씩 반복하면서 각각 10개의 구간에 속하는 지를 체크하여, 속하는 구간의 빈도수를 1씩 증가시킨다.

0 - 99:	10개
100 - 199:	14개
200 - 299:	8개
300 - 399:	7개
400 - 499:	9개
500 - 599:	12개
600 - 699:	9개
700 - 799:	8개
800 - 899:	12개
900 - 999:	11개

[실행 예]