12주차 2차시 구조체

【학습목표】

1. 입출력 이외의 문자열 관련 함수에 대해 설명할 수 있다.
2. 구조체에 대해 설명할 수 있다.

학습내용1 : 입출력 이외의 문자열 관련 함수

1. 문자열의 길이를 반환하는 함수: strlen

```
#include <string.h>
size_t strlen(const char * s);
    ➡ 전달된 문자열의 길이를 반환하되, 널 문자는 길이에 포함하지 않는다.
```

size_t의 일반적인 선언 typedef unsigned int size_t;
typedef에 관해서는 후에 설명

마지막에 삽입되는 널 문자를 없애는 예제

```
void RemoveBSN(char str[])
{
    int len=strlen(str);
    str[len-1]=0;
}

int main(void)
{
    char str[100];
    printf("문자열 입력: ");
    fgets(str, sizeof(str), stdin);
    printf("길이: %d, 내용: %s \n", strlen(str), str);

    RemoveBSN(str);
    printf("길이: %d, 내용: %s \n", strlen(str), str);
    return 0;
}
```

```
문자열 입력: Good morning
길이: 13, 내용: Good morning

길이: 12, 내용: Good morning
```

2. 문자열을 복사하는 함수들: strcpy, strncpy

```
#include <string.h>
char * strcpy(char * dest, const char * src);
char * strncpy(char * dest, const char * src, size_t n);
```

➜ 복사된 문자열의 주소 값 반환

```
int main(void)
{
    char str1[30]="Simple String";
    char str2[30];
    strcpy(str2, str1);
    . . . .    // str1의 문자열을 str2에 복사
}
```

str1에 저장된 문자열을 str2에 단순히 복사!

```
int main(void)
{
    char str1[30]="Simple String";
    char str2[30];
    strncpy(str2, str1, sizeof(str2));
    . . . .
}
```

str1에 저장된 문자열을 str2에 복사하되 최대 sizeof(str2)의 반환 값 크기만큼 복사한다.
strcpy 함수를 호출하는 경우 배열의 범위를 넘어서 복사가 진행될 위험이 있다.

3. strncpy 함수를 잘못 사용한 예

```
int main(void)
{
    char str1[20]="1234567890";
    char str2[20];
    char str3[5];

    /**** case 1 ****/
    strcpy(str2, str1);
    puts(str2);

    /**** case 2 ****/
    strncpy(str3, str1, sizeof(str3));
    puts(str3);

    /**** case 3 ****/
    strncpy(str3, str1, sizeof(str3)-1);
    str3[sizeof(str3)-1]=0;
    puts(str3);
    return 0;
}
```

```
1234567890
12345傲傲傲傲傲?234567890
1234
```

■ strncpy(str3,str1,sizeof(str3)); 문장은 배열 길이 str1에 딱 맞는 길이만큼만 복사를 하겠다는 의도의 문장

■ 두번째 strncpy 함수호출 후의 결과에 이상이 보이는 이유는 복사하는 과정에서 문자열의 끝을 의미하는 널 문자가 복사되지 않았기 때문이다. 문자열을 복사할 때에는 항상 널 문자의 복사까지 고려해야 한다

4. 문자열을 덧붙이는 함수들: strcat, strncat

```
#include <string.h>
char * strcat(char * dest, const char * src);
char * strncat(char * dest, const char * src, size_t n);
```

➜ 덧붙여진 문자열의 주소 값 반환

- strncat 함수는 덧붙일 문자열의 최대 길이를 제한한다.
- 최대 n개의 문자를 덧붙이되 널 문자 포함하여 n+1개의 문자를 덧붙인다.

```
int main(void)
{
    char str1[30]="First~";
    char str2[30]="Second";
    strcat(str1, str2);
    . . . .  // str1의 문자열 뒤에 str2를 복사
}
```

```c
int main(void)
{
    char str1[20]="First~";
    char str2[20]="Second";

    char str3[20]="Simple num: ";
    char str4[20]="1234567890";

    /**** case 1 ****/
    strcat(str1, str2);
    puts(str1);

    /**** case 2 ****/
    strncat(str3, str4, 7);
    puts(str3);
    return 0;
}
```

```
First~Second
Simple num: 1234567
```

5. 문자열을 비교하는 함수들 strcmp, strncmp

```c
#include <string.h>
int strcmp(const char * s1, const char * s2);
int strncmp(const char * s1, const char * s2, size_t n);
```

➜ 두 문자열의 내용이 같으면 0, 같지 않으면 0이 아닌 값 반환

s1이 더 크면 0보다 큰 값 반환
s2가 더 크면 0보다 작은 값 반환
s1과 s2의 내용이 모두 같으면 0 반환

■ strncmp는 최대 n개의 문자를 비교

▶ 크고 작음은 아스키코드 값을 근거로 한다.

▶ A보다 B가, B보다 C가 아스키 코드 값이 더 크고 A보다 a가 , B보다 b가 아스키 코드 값이 더 크니, 사전편찬순서를 기준으로 뒤에 위치할 수록 더 큰 문자열로 인식해도 된다.

```
printf("%d", strcmp("ABCD", "ABCC"));
```
0보다 큰 값이 출력

```
printf("%d", strcmp("ABCD", "ABCDE"));
```
0보다 작은 값이 출력

■ 두 문자열이 같으면 0, 다르면 0이 아닌 값을 반환한다고 인식하고 있어도 충분하다!

6. 문자열 비교의 예

```
int main(void)
{
    char str1[20];
    char str2[20];
    printf("문자열 입력 1: ");
    scanf("%s", str1);
    printf("문자열 입력 2: ");
    scanf("%s", str2);

    if(!strcmp(str1, str2))
    {
        puts("두 문자열은 완벽히 동일합니다.");
    }
    else
    {
        puts("두 문자열은 동일하지 않습니다.");

        if(!strncmp(str1, str2, 3))
            puts("그러나 앞 세 글자는 동일합니다.");
    }
    return 0;
}
```

```
문자열 입력 1: Simple
문자열 입력 2: Simon
두 문자열은 동일하지 않습니다.
그러나 앞 세 글자는 동일합니다.
```

7. 그 외의 변환함수들

```
int atoi(const char * str);        문자열의 내용을 int형으로 변환
long atol(const char * str);       문자열의 내용을 long형으로 변환
double atof(const char * str);     문자열의 내용을 double형으로 변환
```

헤더파일 stdlib.h에 선언

■위의 함수들을 모른다면 문자열에 저장된 숫자 정보를 int형 또는 double형으로 변환하는 일은 번거로운 일이 될 수 있다.

```
int main(void)
{
    char str[20];
    printf("정수 입력: ");
    scanf("%s", str);
    printf("%d \n", atoi(str));
    printf("실수 입력: ");
    scanf("%s", str);
    printf("%g \n", atof(str));
    return 0;
}
```

```
정수 입력: 15
15
실수 입력: 12.456
12.456
```

학습내용2 : 구조체

1. 구조체의 정의

```
int xpos;   // 마우스의 x 좌표
int ypos;   // 마우스의 y 좌표
```

■ 마우스의 좌표정보를 저장하고 관리하기 위해서는 x좌표와 y좌표를 저장할 수 있는 두 개의 변수가 필요하다.
■ xpos와 ypos는 서로 독립된 정보를 표현하지 않고 하나의 정보를 표현한다. 따라서 이 둘은 늘 함께한다.

```
struct point   // point라는 이름의 구조체 정의
{
    int xpos;   // point 구조체를 구성하는 멤버 xpos
    int ypos;   // point 구조체를 구성하는 멤버 ypos
};
```

■ 구조체를 이용해서 xpos와 ypos를 하나로 묶었다. 이 둘을 묶어서 point라는 이름의 새로운 자료형을 정의
■ int가 자료형의 이름인것 처럼 point도 자료형의 이름이다. 단, 프로그래머가 정의한 자료형이기에 '사용자 정의 자료형(user defined data type)'이라 한다.

```
struct person
{
    char name[20];      // 이름 저장
    char phoneNum[20];  // 전화번호 저장
    int age;    // 나이 저장
};
```

개인의 이름과 전화번호 나이 정보를 person이라는 구조체 정의를 통해서 묶고 있다.
■ 배열도 구조체의 멤버로 선언이 가능

2. 구조체 변수의 선언과 접근

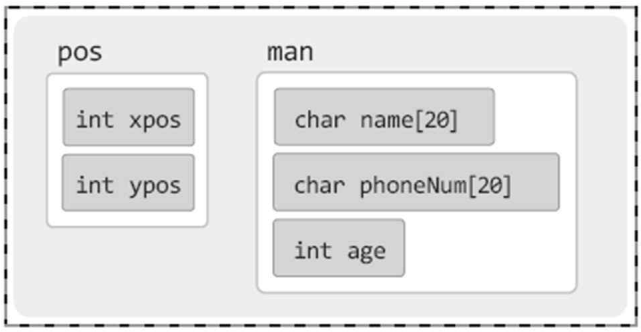
■ 구조체 변수선언의 기본 형태

struct *type_name val_name ;*

struct point pos;

struct person man;

구조체 변수선언의 예



구조체 변수선언의 결과

■ 멤버의 접근방식

구조체 변수의 이름 .구조체 멤버의 이름

pos.xpos=20;

구조체 변수 *pos*의 멤버 *xpos*에 *20*을 저장

printf("%s \n", man.name);

*man*의 멤버 *name*에 저장된 문자열 출력

3. 구조체 변수의 선언과 접근 관련 예제1

```
struct point    // 구조체 point의 정의
{
    int xpos;
    int ypos;
};

int main(void)
{
    struct point pos1, pos2;
    double distance;
    fputs("point1 pos: ", stdout);
    scanf("%d %d", &pos1.xpos, &pos1.ypos);
    fputs("point2 pos: ", stdout);
    scanf("%d %d", &pos2.xpos, &pos2.ypos);

    /* 두 점간의 거리 계산 공식 */
    distance=sqrt((double)((pos1.xpos-pos2.xpos) * (pos1.xpos-pos2.xpos)+
            (pos1.ypos-pos2.ypos) * (pos1.ypos-pos2.ypos)));
    printf("두 점의 거리는 %g 입니다. \n", distance);
    return 0;
}
```

```
point1 pos: 1 3
point2 pos: 4 5
두 점의 거리는 3.60555 입니다.
```

이 예제에서 호출하는 함수 sqrt는 제곱근을 반환하는 함수로써 헤더파일 math.h에 선언된 수학관련 함수이다

4. 구조체 변수의 선언과 접근 관련 예제2

```
struct person
{
    char name[20];
    char phoneNum[20];
    int age;
};
int main(void)
{
    struct person man1, man2;
    strcpy(man1.name, "안성준");
    strcpy(man1.phoneNum, "010-1122-3344");
    man1.age=23;
    printf("이름 입력: "); scanf("%s", man2.name);
    printf("번호 입력: "); scanf("%s", man2.phoneNum);
    printf("나이 입력: "); scanf("%d", &(man2.age));
    printf("이름: %s \n", man1.name);
    printf("번호: %s \n", man1.phoneNum);
    printf("나이: %d \n", man1.age);
    printf("이름: %s \n", man2.name);
    printf("번호: %s \n", man2.phoneNum);
    printf("나이: %d \n", man2.age);
    return 0;
}
```

```
이름 입력: 김수정
번호 입력: 010-0001-0002
나이 입력: 27
이름: 안성준
번호: 010-1122-3344
나이: 23
이름: 김수정
번호: 010-0001-0002
나이: 27
```

구조체의 멤버라 하더라도 일반적인 접근의 방식을 그대로 따른다.
구조체의 멤버로 배열이 선언되면 배열의 접근방식을 취하면 되고, 구조체의 멤버로 포인터 변수가 선언되면 포인터 변수의 접근방식을 취하면 된다.

5. 구조체 정의와 동시에 변수 선언하기

```
struct point
{
    int xpos;
    int ypos;
} pos1, pos2, pos3;
```

point라는 이름의 구조체를 정의함과 동시에 point 구조체의 변수 pos1, pos2, pos3를 선언하는 문장이다.

```
struct point
{
    int xpos;
    int ypos;
};
struct point pos1, pos2, pos3;
```

위와 동일한 결과를 보이는 구조체의 정의와 변수의 선언이다.

▪구조체를 정의함과 동시에 변수를 선언하는 문장은 잘 사용되지 않는다. 그러나 문법적으로 지원이 되고 또 간혹 사용하는 경우도 있다.

6. 구조체 변수의 초기화

```
struct point
{
    int xpos;
    int ypos;
};
struct person
{
    char name[20];
    char phoneNum[20];
    int age;
};
int main(void)
{
    struct point pos={10, 20};
    struct person man={"이승기", "010-1212-0001", 21};
    printf("%d %d \n", pos.xpos, pos.ypos);
    printf("%s %s %d \n", man.name, man.phoneNum, man.age);
    return 0;
}
```

```
10 20
이승기 010-1212-0001 21
```

초기화 방식이 배열과 유사하다.
초기화 할 데이터들을 중괄호 안에 순서대로 나열하면 된다,

【학습정리】

1. 모니터로 하나의 문자를 출력할 때 일반적으로 사용하는 두 함수는 putchar, fputc 이고, 키보드로부터 하나의 문자를 입력받을 때 일반적으로 사용하는 두 함수는 getchar, fgetc이다.

2. 문자단위의 입출력 함수가 존재하는 이유는 printf, scanf함수를 사용할 경우 메모리 사용 공간도 크고, 연산도 많아서 상대적으로 속도가 느리기 때문에 단순히 문자 하나를 입출력 할 경우 문자단위의 입출력 함수를 쓰는 것이 좋다.

3. 모니터로 하나의 문자열을 출력할 때 일반적으로 사용하는 두 함수는 puts, fputs이다.

4. 구조체란 하나 이상의 변수(포인터 변수와 배열 포함)를 묶어서 새로운 자료형을 정의하는 도구이다.