c언어 - HW1

임베디드스쿨1기

Lv1과정

2020. 04. 23

이충재

* 배열 0으로 초기화하기: 자료형 배열명[크기] = { 0, };

ex) int arr[10] = { 0, }; → arr[0] ~ arr[9] 전부 0이 된다.

* 배열을 포인터에 넣기

배열은 첫번째 요소의 주소를 가진다.
배열도 주소 값이기 때문에 포인터에 넣을 수 있다.

ex) int arr[3];
int* pointer = arr;

→ pointer 포인터에 arr 배열을 넣었다.

포인터에 이중배열 넣기

```
Int array[3][3];
Int** ppointer = array;
```

이중배열을 이중포인터에 넣을 수 있을것 같지만
실제로 프로그램을 실행해보면 안된다.

```
Int array[2][3];
Int (*pointer)[3] = array;
```

포인터에 이중배열을 넣기 위해서는 다음과 같은
특별한 방법으로 포인터를 만들어야한다.

왼쪽 예시는 두번째 인덱스의 크기가 3인 배열을
가리키는 포인터를 나타낸다.

* 포인터에 동적 메모리 할당하기

malloc함수를 사용하면 메모리를 할당할수 있다.

ex) int* pointer = malloc(sizeof(int) *4);

위 예시는 pointer포인터에 int자료형의 크기의 4배만큼 메모리를 할당하였다.

포인터를 배열처럼 사용하기.

포인터에 동적메모리를 할당하면 배열처럼 사용 할 수 있다.

ex) int* p = malloc(sizeof(int) * 4);
P[0] = 1;
p[1] = 2;

왼쪽 예시는 포인터p에 int크기의 공간 4개를 만들고
첫 번째 공간에 1, 두 번째 공간에 2를 넣은 것과 같다.

* 포인터에 문자열 저장하기

Char포인터에 문자열을 큰따옴표에 넣어 할당 할 수 있다.

ex) char* s1 = “Hello”; → 포인터 s1에 “Hello”문자열이 저장된 주소를 저장

포인터에 직접 문자열을 저장하는 것이 아닌 문자열이 저장된 주소를 저장한다.

문자열 포인터에서 인덱스로 접근 가능하다.

```
Char* s1 = “Hello”;
Printf(“%c”, s1[0]);
```

주의할점: 문자열 포인터는 인덱스로 문자 할당을 하지 못한다.

~~ex) char*s1 = “Hello”;~~
~~s1[0] = A;~~

* 배열 형태로 문자열 저장하기

ex) char s1[10] = “Hello”;

크기가 10인 배열에 문자열 “Hello”넣었다.
이때 메모리 공간 하나에 문자 한 개씩 저장된다.
저장하고 남은 공간에는 NULL문자로 채워진다.

*주의점: 문자열 끝에는 항상 NULL문자가 붙는다.
따라서 배열에 문자열을 저장 할 때 배열 크기가
문자열 크기보다 최소 +1 더 커야한다.

구조체: 자주 사용되는 동일한 성격을 가진 변수를 한 곳에 모아둔것.

구조체 생성방법 : struct 구조체이름 { 구조체 멤버 설정};

구조체 멤버: 구조체내 사용된 변수

```
ex)  struct person
     {
        char name[20];
         int age;
         char address[100];
     };
```

개인정보라는 성격을 가진 변수들을
person 구조체에 모아놨다.

구조체 마지막에 세미콜론 필요

구조체 변수: 구조체멤버에 접근하기 위한 변수

```
Int main(void)
{
  struct person p1;
  p1.age = 30;
}
```

구조체 변수 선언: struct 구조체명 변수명
왼쪽 예시에서 person이 구조체명, p1이 변수명이다.

구조체변수와 구조체멤버 사이 점(.)을 찍어
구조체 person의 멤버 age에 접근하였다.

구조체 포인터: 구조체 멤버에 접근하기위한 포인터.

```
Int main(void)
{
        struct person *p;
        P → age = 30;
}
```

Struct 구조체명 *구조체포인터명
구조체포인터이름 앞에 *붙이는것 주의!

화살표로 구조체 멤버에 접근한다.

typedef struct 구조체이름 새로운이름
struct 구조체이름 → 새로운 이름으로 사용

ex) typedef struct person per;
struct person을 per로 대체가능

struct person p1; → Per p1;
구조체 변수선언시 조금 더 간단히 할 수 있다.

구조체배열: 여러개의 구조체변수를 배열에 담은것.

구조체 변수가 많을때 유용.
ex)struct person p1, struct person p2, …. p100

struct person p[100];

p[100]

p1	p2	p3	(중)	p98	p99	p100

p[0] p[1] p[2] * * * p[97] p[98] p[99]

구조체 배열도 마찬가지로 구조체 멤버접근시 . 이용한다.

ex) p[0].name
P[1].address

구조체 포인터 배열: 구조체 포인터를 담은 배열

Struct person *p[100];

*p1	*p2	*p3	(중략)	*p98	*p99	*p100

P[0] p[1] p[2] * * * p[97] p[98] p[99]

구조체 포인터와 마찬가지로 ->로 멤버에 접근한다.

구조체 포인터를 구조체배열처럼 사용하기.

Struct person* p = malloc(sizeof(struct person) * 64)
=> 구조체변수 64개크기 만큼 동적메모리할당.
=> 구조체변수 64개 만든것과 동일효과

멤버 접근시 구조체배열과 마찬가지로 P[0].name과 같이 사용 할 수 있다.

구조체크기: 자료형이 가장 큰 것이 결정한다.
가장 큰 자료형이 기본단위가 된다.

ex) struct str
{
char s1; ← 1바이트
int num1; ← 4바이트
};

int가 가장 크기 때문에 4바이트가 기본단위이다.
구조체의 크기는 4바이트(char) + 4바이트(int)
= 8바이트이다.

ex) struct str
{
char name[10]; ← 10바이트
int age; ← 4바이트
};

int가 가장 크기때문에 4바이트가 기본단위이다.
구조체크기 = name(12바이트) + age(4바이트)
= 16바이트이다.

구조체 변수 크기: 구조체변수의 크기는 구조체크기와 같다.
구조체 멤버의 크기와 수에 따라서 구조체변수 크기 달라진다.

구조체포인터 크기: 포인터의 크기와같다.(8바이트)