새싹교실

Week5

- Pointer review
- C Language (배열과 포인터)

고주형

2019/05/22

Review

Review

- &: 주소 연산자
- pointer: 주소를 저장하는 곳(변수)

Review

주소는 그냥 정수인데 왜 int형에 저장하지 않고 pointer에 저장할까?

1. 어떻게 쓸지 모른다.
2. 그 메모리로부터 몇 바이트를 쓸지 모른다.

C Programming

오늘 할 것

- Double Pointer
- Array
- Pointer and Array
- How to use Pointer
- Dynamic Allocation

Double Pointer

Pointer를 저장하면 특정 메모리(주소)에 저장될 것이다.

그러면 Pointer의 메모리 주소를 저장 위한 자료형은? Double Pointer!

Array

같은 형태(자료형)의 값을 여러 번 저장하고 싶을 땐?

Array

1. 1차원 a[크기]
2. 2차원 a[행][열]
3. 3차원 a[높이][행][열]
.
.
.
n. n차원

Array 초기화

0,만 넣어도 모두 0으로 초기화된다.
Ex) a[99][99][999] = {0,};

Pointer and Array

1. 1차원 배열과 포인터
   int arr[] = {1,2,3};

arr은 사실은 첫번째 원소를 가리키는 1차원 포인터이다
(주의) 일반적인 변수처럼 arr의 값은 수정될 수 없다.
   Ex) arr = &b; //에러남!!!
그 외의 포인터의 연산은 먹는다.
   Ex) printf("%d",*arr); // 1 출력

Quiz

1. arr[2]를 포인터 연산을 사용하여 나타내보자.

Pointer and Array

```
int arr[] = {1,2,3};
```

arr은 바뀔 수 없는 1차원 포인터라고 설명했다.
포인터는 당연히 포인터 변수에 저장할 수 있으므로
arr은 포인터에 저장할 수 있다.
Ex)

```
int* arrCpy = arr;
printf("%d",arrCpy[2]);
printf("%d",*arrCpy);
```

다음의 출력 결과를 예상해 보자.

arr과 arrCpy의 차이점

arr과 arrCpy의 차이점은 뭘까?
그냥 배열과 포인터는 같은가?

크기가 다르게 인식된다는 차이가 있다.
- 배열의 경우 처음에 크기를 알려줬기 때문에(정적)
그 것을 기반으로 크기가 계산이 된다.
- 배열의 주소를 저장한 포인터의 경우 그냥 일반적
인 포인터 변수 인식되어서 포인터의 크기가 나온다.

arr과 arrCpy의 차이점

sizeof(arr)과 sizeof(arrCpy)의 결과를 확인해보자.

Quiz

pointer의 크기는 전부 다를까?
다음의 결과를 확인해보자.
sizeof(char*) sizeof(int*) sizeof(long*)
sizeof(float*) sizeof(double*)

다 같을 것이다.
32비트 운영체제 포인터 크기=>4바이트(32비트)로 동일
64비트 운영체제 포인터 크기=>8바이트(64비트)로 동일

Pointer and Array

2. arr2[][] {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12},
}

2차원 배열은 그럼 더블 포인터에 복사하여 이용할 수 있을까?

Ex) int** ptr = arr2;
    printf("%d", ptr[1][2]); //안된다…

Pointer and Array

그러면 어떻게 이중 포인터에 2차원 배열을 저장할 수 있을까? 좀 더 다른 방법을 생각해보자.

그러기 위해서 배열 포인터와 포인터 배열에 대해 알아보자.

Pointer and Array

배열 포인터: int (*ptr)[열의 크기]
 - 배열들을 가리키는 포인터들을 저장

포인터 배열: int *ptr[저장할 포인터의 수]
 - 포인터들을 저장하기 위한 배열
    Ex) int *ptr[] = {&a,&b,&c};

Pointer and Array

이차원 배열을 배열 포인터에 저장해보자.

```
int (*arrPtr)[4] = arr2;
```

(주의)

저장할 배열과 배열 포인터의 **열의 크기**가 같아야 한다.

Quiz

sizeof(arr2)와 sizeof(arrPtr)의 차이는?

복사된 arrPtr을 배열처럼 생각하고 사용해보자!
Ex) arrPtr[1][2]

Dynamic Allocation Intro

1. 배열의 한계
- 배열은 정적이다.
- int arr[변수];//Error!!!
2. 동적으로 사용하려면 malloc하자.
 - int* newArr = (int*)malloc(sizeof(int)*크기);
 - malloc을 사용하려면 #include <stdlib.h>를 상단에 해줘야 한다.

Dynamic Allocation

malloc한 것을 2차원 배열처럼 사용하려면?
a. double pointer <=배열의 세로축 할당
int** arr2 = malloc(sizeof(int*)*세로 크기);

b. arr2[i]에 가로축 할당
for(int i=0;i<세로 크기;i++){
arr2[i] = malloc(sizeof(int) * 가로 크기);
}

Dynamic Allocation

마지막으로 동적할당을 해줬으면 free하는 것을 잊지 말자

free(해제할 주소);

포인터 실습

1. 2차원 배열 익숙해지기
   int arr[3][1];

a. 이 배열의 가로 크기를 sizeof()를 사용하여 구해보자.

b. 이 배열의 세로 크기를 sizeof()를 사용하여 구해보자.

Hint) 가로축의 size를 이용하자. <= sizeof(arr[0]); 이용

답

sizeof(arr[0]) = sizeof(int) * 가로크기
따라서, 가로크기 = sizeof(arr[0])/sizeof(int);

sizeof(arr) = sizeof(int) * 가로크기 * 세로크기
따라서, 세로크기 = sizeof(arr)/sizeof(arr[0]);

Next Week

문자열의 “ ”, ‘ ’의 차이

사용자 정의 자료형 - 3차원 벡터 만들어 보기

구조체, 구조체와 포인터, 문자열, typedef