포인터

주소값의 이해

- 데이터의 주소값이란 해당 데이터가 저장된 메모리의 시작 주소를 의미합니다.
- C언어에서는 이러한 주소값을 1바이트 크기의 메모리 공간으로 나누어 표현합니다.
- int형 데이터는 4바이트의 크기를 가지지만, int형 데이터의 주소값은 시작 주소 1바이트만을 가리킵니다.

포인터란?

- C언어에서 포인터(pointer)란 메모리의 주소값을 저장하는 변수이며, 포인터 변수라고도 부릅니다.
- char형 변수가 문자를 저장하고, int형 변수가 정수를 저장하는 것처럼 포인터는 주소값을 저장합니다.

```
int n = 100;   // 변수의 선언
int *ptr = &n; // 포인터의 선언
```

> C언어에서 포인터와 연관되어 사용되는 연산자

1. 주소 연산자(&)

- 주소 연산자는 변수의 이름 앞에 사용하여, 해당 변수의 주소값을 반환합니다.
- '&'기호는 앰퍼샌드(ampersand)라고 읽으며, 번지 연산자라고도 불립니다.

2. 참조 연산자(*)

- 포인터의 이름이나 주소 앞에 사용
- 포인터에 가리키는 주소에 저장된 값을 반환

포인터의 선언

```
타입* 포인터이름;
```

- 타입이란 포인터가 가리키고자 하는 변수의 타입을 명시합니다.
- 포인터 이름은 포인터가 선언된 후에 포인터에 접근하기 위해 사용됩니다.

> 포인터의 선언과 동시에 초기화를 함께 하는 것이 좋습니다.

```
타입* 포인터이름 = &변수이름;
타입* 포인터이름 = 주소값;
```

포인터의 참조

> 다음 예제는 포인터의 주소값과 함께 포인터가 가리키고 있는 주소값의 데이터를 참조하는 예제입니다.

```
int x = 5;         // 변수의 선언
int *ptr = &x;     // 포인터의 선언
int *pptr = &ptr; // 포인터의 참조
```

포인터와 배열

> 포인터를 사용하여 배열에 접근할 수도 있습니다.

> C에서 배열의 이름은 실제로 배열의 첫 번째 요소 에 대한 포인터 입니다

```
int myNums[4] = {25, 50, 75, 100};
// myNums의 첫 번째 요소 값을 가져옵니다.
printf("%d", *myNums);
결과
25
```

> myNums의 나머지 요소에 접근하려면 포인터/배열(+1, +2)을 증가시킬 수 있습니다.

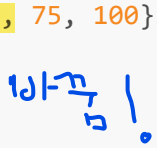
```
                [0] [1] [2] [3]
int myNums[4] = {25, 50, 75, 100};
// myNums의 두 번째 요소 값을 가져옵니다.
printf("%d\n", *(myNums + 1));
// myNums의 세 번째 요소 값을 가져옵니다.
printf("%d", *(myNums + 2));
결과
50
75
```

> 반복문을 사용해서 확인하자

```
int myNums[4] = {25, 50, 75, 100};
int *ptr = myNums;
int i;

for (i = 0; i < 4; i++) {   0 1 2 3
  printf("%d\n", *(ptr + i));
}
결과
25
50
75
100
```

> 포인터를 사용하여 배열 요소의 값을 변경하는 것도 가능합니다.

```
int myNums[4] = {25, 50, 75, 100};

*myNums = 13;
*(myNums + 1) = 17;

printf("%d\n", *myNums);
printf("%d\n", *(myNums + 1));
```



결과

```
13
17
```

> 문자열은 실제로 배열이므로 포인터를 사용하여 문자열 에 접근할 수도 있습니다 .