C언어의 꽃 ..... << 포 인 터 >>

자료형 *포인터이름;

포인터 = &변수;

```c
#include <stdio.h>

int main()
{
    int *numPtr;      // 포인터 변수 선언
    int num1 = 10;    // int형 변수를 선언하고 10 저장

    numPtr = &num1;   // num1의 메모리 주소를 포인터 변수에 저장

    printf("%p\n", numPtr);    // 0055FC24: 포인터 변수 numPtr의 값 출력
                               // 컴퓨터마다, 실행할 때마다 달라짐
    printf("%p\n", &num1);     // 0055FC24: 변수 num1의 메모리 주소 출력
                               // 컴퓨터마다, 실행할 때마다 달라짐

    return 0;
}
```

실행결과

```
0055FC24 (메모리 주소. 컴퓨터마다, 실행할 때마다 달라짐)
0055FC24 (메모리 주소. 컴퓨터마다, 실행할 때마다 달라짐)
```

위 코드를 잘 이해하는게 중요함

1. int *numPtr -> 이게 포인터 변수를 선언하는 것
2. int num1 -> int형 변수를 선언하고 10 저장

1번이랑 2번이랑 같다고 생각하면 되는데 자료형 구조가 다른거 뿐임

numPtr = &num1;

이 부분을 확실히 개념을 잡고 들어가야함

numPtr은 포인터 변수형이고 &num1은 10이 저장된 주소값을 뜻함

즉 numPtr에 10이 저장된 주소값을 저장시킨 것

그럼 numPtr을 출력시키면 어떤 값이 나올까 ??

```
참고용
32비트와 64비트의 포인터 크기

32비트인 경우에는 포인터 자료형 크기는 4byte
64비트인 경우에는 포인터 자료형 크기는 8byte
```

이제부터 어려워지기 시작함

배열의 포인터

배열도 자료형이기 때문에 당연히 배열을 위한 포인터 또한 존재

```
int a[10]; // int [10] 자료형 변수 a. 크기는 40 바이트
int (*p)[10] = &a; // int [10] 자료형의 포인터 변수 p. p의 자료형은 int (*)[10]

(*p)[3] = 3; // a[3] = 3 과 같음
*p[3] = 3; // a[30] = 3 과 같음. 오버플로우!

int *pa = a; // int 자료형의 포인터 변수 pa.
pa[3] = 3; // a[3] = 3과 같음
```

변수 p는 변수 a의 주소값을 가지고 있는 포인터 변수이다.

변수 p를 통해서 a에 접근할 때는 어떤 방법으로 접근이 가능할까 ?

- *p 또는 p[0]을 적어주면 된다.

*p는 주소값이기 때문에 a 배열의 있는 원소 값들에 접근이 가능하다.

3번째에 있는 값을 접근할때는

(*p)[3] 이렇게 접근을 해야한다. 만약에 *p[3]이렇게 접근하면 어떤 현상이 발생할까 ?? (숙제)

포인터의 포인터

포인터 또한 자료형이기 때문에 마찬가지로 포인터를 위한 포인터도 정의 가능

```
int a; // int형 변수 a. 크기는 4 바이트
int *pa = &a; // int * 자료형 pa. 크기는 8 바이트
int **ppa = &pa; // int ** 자료형 ppa. 크기는 8 바이트

int b[10]; // int [10] 자료형 b. 크기는 40 바이트
int (*pb)[10] = &b; // int (*)[10] 자료형 pb. 크기는 8 바이트
int (**ppb)[10] = &pb; int (**)[10] 자료형 ppb. 크기는 8 바이트

int *c[10]; // int *[10] 자료형 c. 크기는 80 바이트
int *(*pc)[10] = &c; // int *(*)[10] 자료형 pc. 크기는 8 바이트
```

약간 좀 복잡하다고 생각할수 있는데 단순하게 생각해보면

a의 주소값 -< pa에 저장

pa에 저장하는 주소가 또 존재하겠죠 ... 그럼 이 존재하는 주소값을 다시 한번 포인터로 접근하는게 포인터의 포인터

동적할당

프로세스의 실행중에 메모리를 할당하는 것 !

배열의 크기를 변수에 의해 결정할 때 자주 사용한다!

malloc이라는 함수를 많이 들어봤을텐데 이 함수는 동적할당을해주는 함수이다.

```
// int 형 변수 동적할당
int *a = malloc(sizeof(int));

// 1차원 배열의 동적할당
int *b = (int *)malloc(sizeof(int) * N);
```

2차원 배열을 동적할당하는거 ...

```
// 2차원 배열 N * M의 동적할당

int **a = (int **)malloc(sizeof(int *) * N);
for(int i=0; i < N; i++)
    a[i] = (int *)malloc(sizeof(int) * M);
```

어렵...

각 자료형에대한 크기를 계산하는게 중요하다 연습하자