포인터 활용

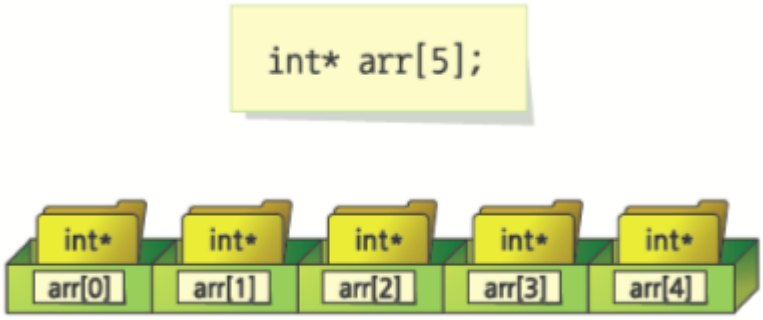
1. 포인터 배열

▼ 포인터 배열 : 주소를 저장하는 배열

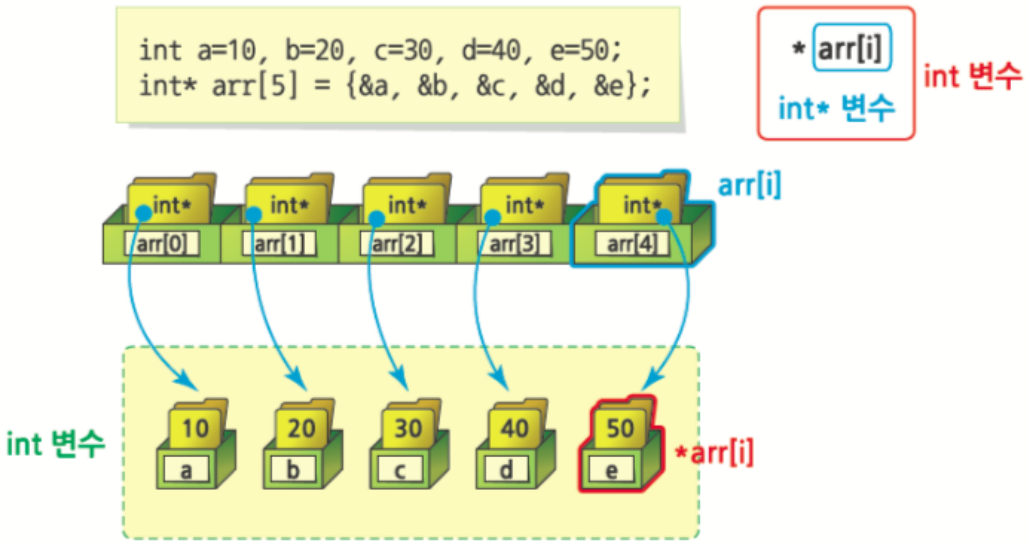
▼ 포인터 배열 형식

```
데이터형* 배열명[배열크기];

int* arr1[5];
char* arr2[10];
double* arr3[4];
PRODUCT* arr4[3];
```

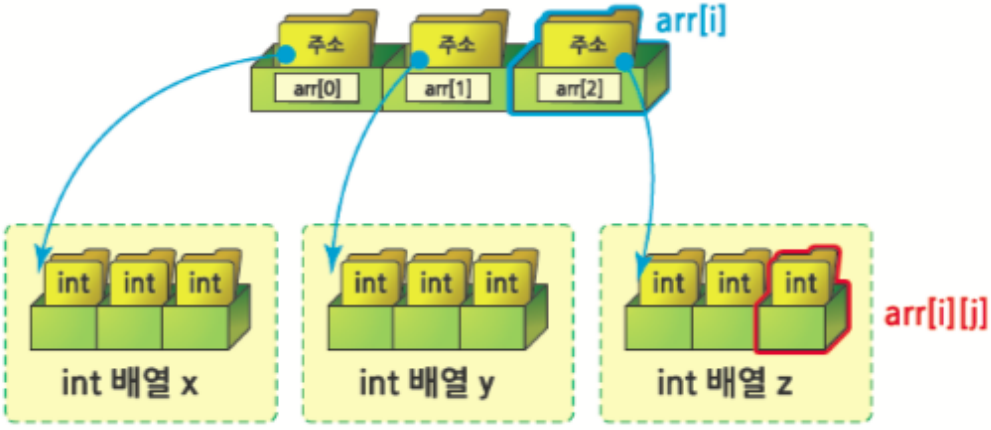


▼ 포인터 배열의 값을 사용하려면 간접 참조 연산자 *를 사용해야 한다. (ex1101.c)



▼ 포인터 배열의 각 원소에 다른 배열의 시작 주소를 저장 할 수 있다. (ex1102.c)

이 때, 다른 배열의 값은 arr[i][j] (= *(arr[i]+j)) 와 같이 접근할 수 있다.

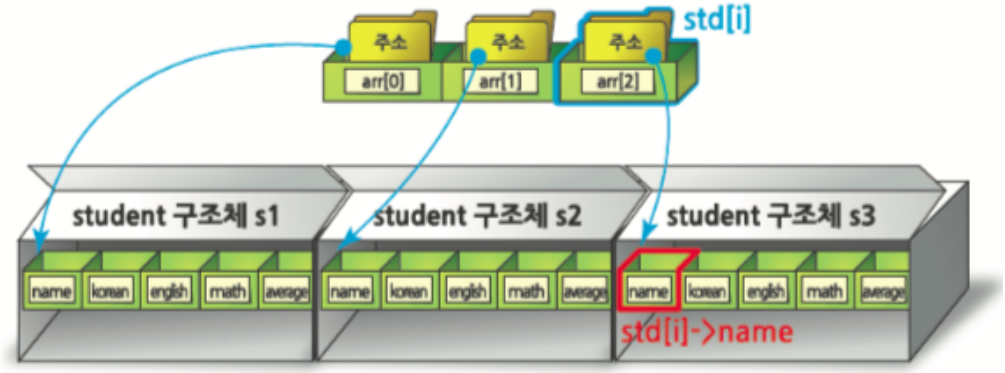


▼ 구조체 포인터 배열

- 구조체 포인터 배열은 메모리 절약을 위해 자주 사용 된다.
- 예를 들어, 구조체가 28 byte 사용할 경우, 크기 100 인 구조체 배열을 선언 ⇒ 2800 byte
- 구조체 포인터 배열의 경우 주소 4byte 만 사용하므로 ⇒ 400 byte

```
PRODUCT prd[100]; => 2800 byte
PRODUCT* prd[100]; => 400 byte
```

- PRODUCT* prd[100]에서 prd의 원소는 PRODUCT* 형이므로 → 사용으로 멤버 접근
- ex1103.c 예제 확인



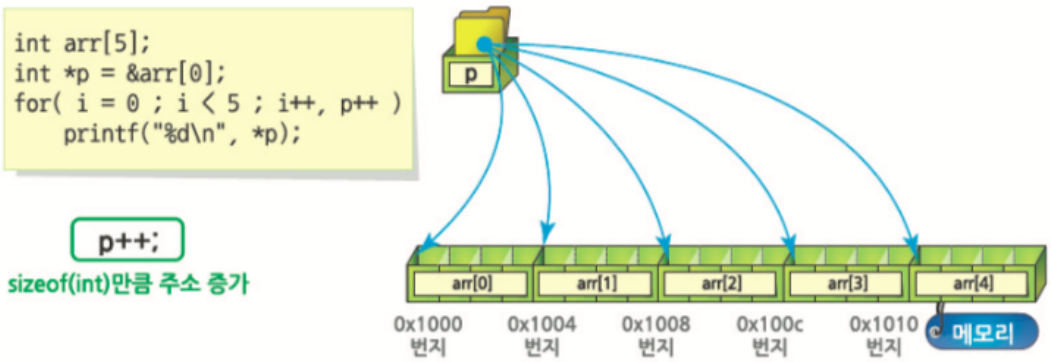
2. 배열에 대한 포인터

▼ 배열 전체를 가리키는 포인터를 배열에 대한 포인터라 한다. ⇒ 2차원 배열부터 의미가 있다.

▼ 배열에 대한 포인터 형식

```
데이터형 (*포인터명)[배열크기];

int (*p)[5];
...
int arr[3][5];
int (*p)[5] = arr;
```

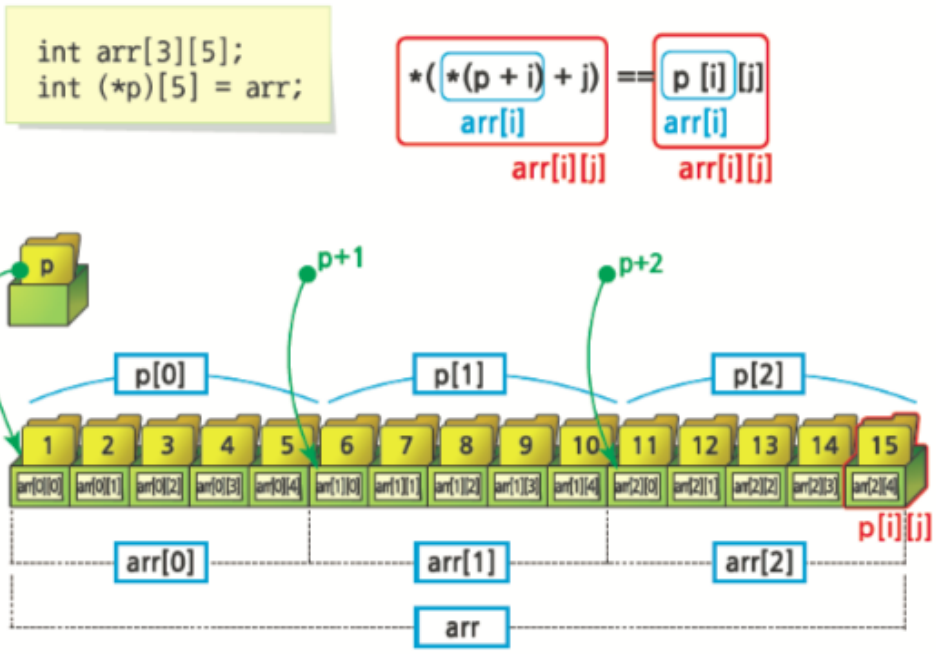


i = 0일 때, p는 0x1000번지이고, *p는 arr[0]입니다.
i = 1일 때, p는 0x1004번지이고, *p는 arr[1]입니다.
i = 2일 때, p는 0x1008번지이고, *p는 arr[2]입니다.
i = 3일 때, p는 0x100c번지이고, *p는 arr[3]입니다.
i = 4일 때, p는 0x1010번지이고, *p는 arr[4]입니다.

▼ 배열에 대한 포인터에서 변수의 값 접근

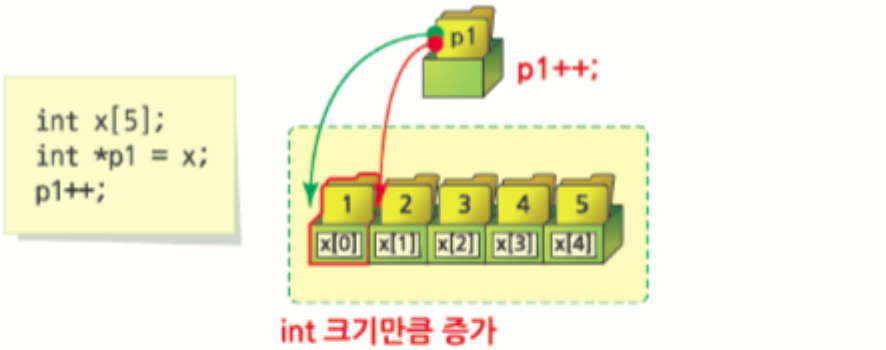
- 포인터가 배열을 가리키는 경우 배열의 인덱스 사용과 마찬가지로 p[i], p[i][j] 등을 사용할 수 있다.

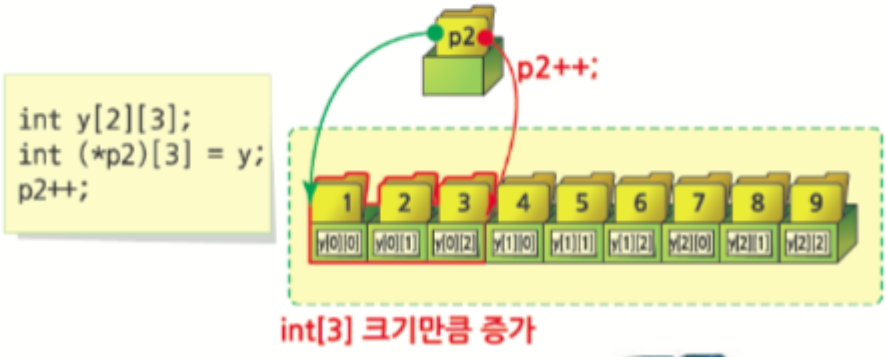
- 즉, 포인터가 배열의 원소를 가리키는 경우 p[i]
- 포인터가 배열 전체를 가리키는 경우 p[i][j] 를 사용
- int (*p)[5] 에서 p[i][j] ⇒ p[i] 로 찾은 int[5] 배열의 j 번째 원소 (그림 참조)



- 이때, p[i][j] = * (*(p+i) + j) 이다.
- ex1104.c 예제 확인

▼ 배열의 원소를 가리키는 포인터와 배열에 대한 포인터의 비교 (ex1105.c)





- array_pointer.c 참조

3. 함수에 대한 포인터

▼ 함수에 대한 포인터 = 함수의 주소를 저장하는 포인터

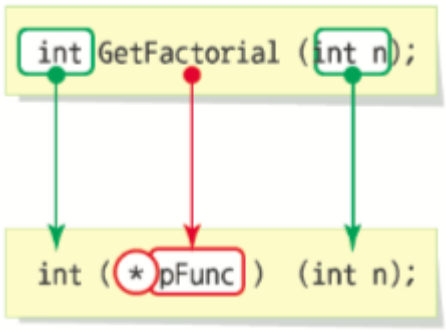
함수도 컴파일 및 링크 후에 메모리의 특정 번지에 할당되므로 주소를 가져올 수 있다.

▼ 함수에 대한 포인터 형식

```
리턴형 (*포인터명)(매개변수);

int (*pFunc)(int);
char (*p2)(double, double);
```

- 함수에 대한 포인터를 정의하기 위해서는 함수의 원형이 필요하다.



- 함수의 원형이 int GetFactorial(int n); 일 경우, 포인터는 int (*pFunc)(int n); 과 같이할 수 있다.
- 초기화 : int (*pFunc)(int n) = GetFactorial; ⇒ 함수명이 함수의 시작 주소를 나타낸다.
- 쓰레기 값으로 초기화 되는 것을 피하기 위해 int (*pFunc)(int n) = NULL; 로 초기화가 가능하다

▼ 포인터를 이용한 함수 호출

- 함수 호출 형식 (ex1106.c)

```
(*포인터명)(인자);
포인터명(인자);

(*pFunc)(10);
pFunc(10);
```

- 함수 호출시 (*pFunc)(10)에서 ( )를 생략하면 error 발생

▼ 함수에 대한 포인터형

- typedef 를 이용해 함수에 대한 포인터형을 정의할 수 있다.
- 형식

```
typedef 리턴형 (*포인터형명)(매개변수);

typedef int (*pFUNCPTR)(int);
typedef double (*FP)(double, double);
```

- 함수에 대한 포인터를 정의하는 형식 앞에 typedef 를 붙여서 함수에 대한 포인터형을 정의한다.
- 함수에 대한 포인터형 변수의 선언과 초기화, 호출

```
typedef int (*FUNCPTR)(int);  // 정의

FUNCPTR pFunc = NULL;  // pFunc 변수 선언과 초기화

pFunc = GetFactorial // 함수의 주소 할당

pFunc(5) // 함수 호출
```

-

4. 실습