포인터

주소연산자

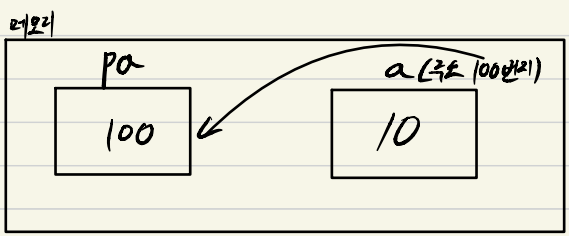
주소: 변수가 할당된 메모리의 시작 주소
&: 주소연산자, 변수의 주소 값을 구함

ex) int a;
printf("%u", &a); // a의 주소 값을 10진수로 출력
printf("%p", &a); // 16진수로 출력

포인터

pointer (포인터): 메모리의 주소값을 저장하는 변수

ex) int a = 10; // 일반변수 a
int* pa; // int*형 포인터 변수 pa 선언
pa = &a; // 포인터 변수 pa에 a의 주소 값 대입



⇒ pa는 변수 a가 어디에 할당 되었는지 알고있다.

pa ⟶ a

: 포인터 변수 pa가 a를 가리킨다!

X int* pa ⟶
↳ int형 변수를 가리키는 포인터 변수 pa

간접참조 연산자 *

↳ 포인터 변수가 가리키는 변수를 사용하게 해줌

ex) int a = 10; // 일반변수 a
int* pa; // int*형 포인터 변수 pa 선언
pa = &a; // 포인터 변수 pa에 a의 주소 값 대입
*pa = 20;
↳ 포인터 변수 pa가 가리키는 변수의 값에 20 대입

※ int* pa;
↳ 연산자 아님

*pa = 20;
↳ 연산자임

즉 *pa == a

Const 포인터

Const 예약어를 포인터 변수에 사용
↳ 가리키는 변수의 값을 바꿀 수 없다는 의미

ex) int a = 10, b = 20;
const int* pa = &a; // pa → a
pa = &b; // 가능하다.
*pa = 20; // Error
↳ pa가 가리키는 변수 a의 값을 바꿀 수 없다.

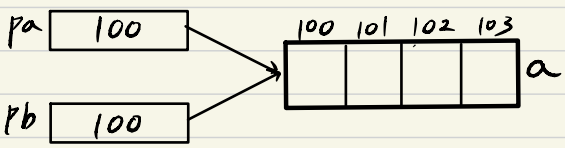
a = 20; // 가능하다.

주소와 포인터의 차이

주소는 '상수'고 포인터는 '변수'다.

⇒ 두 포인터가 같은 주소를 저장 가능하다.

ex) int a;
int *pa, *pb;
pa = pb = &a;



&a = &b // 주소 값은 상수이다.

Error!!

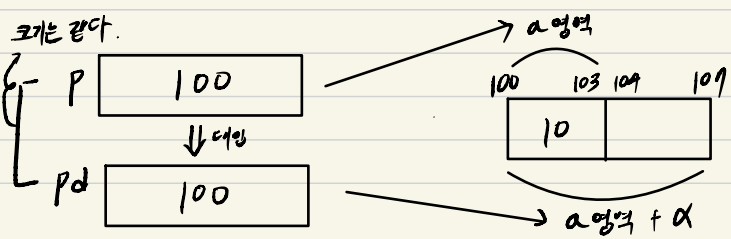
주소와 포인터의 크기

★ 모든 주소와 포인터는 가리키는 자료형에 관계없이 크기가 같다 ★

포인터의 대입

-포인터는 가리키는 변수의 형태가 같을 때만 대입한다.

ex) int a = 10;
int* p = &a;
double* pd;
pd = p; // 경고 메세지 표시



- 형 변환을 사용한 포인터 대입은 언제나 가능하다.

ex)
```
double a = 3.4;
double* pd = &a;
int* pi;
pi = (int*)pd;
```

포인터의 효과

함수 간에 데이터 공유가 가능하다.

ex) main()에서 a = 10 으로 초기화된 변수 a의 값을
swap()이라는 함수에서 메모리에 직접 접근하여 바꿀 수 있다.