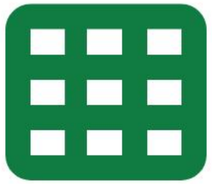


포인터 완전 정복

복잡한 메모리 주소? **"엑셀(Excel)"** 하나면 끝!

메모리는 거대한 엑셀 시트다

컴퓨터의 메모리(RAM)를 어렵게 생각하지 마세요.

그냥 **행 번호(주소)**가 붙어있는
아주 긴 엑셀 시트일 뿐입니다.

- ✓ **행 번호** = 주소 (Address)
- ✓ **셀 내용** = 값 (Value)
- ✓ **이름 상자** = 변수명 (Name)

	A	B
1001	a **10**	
1002		
...	...	...
2005	p **1001**	

일반 변수 vs 포인터 변수

일반 변수

```
int a = 10;
```

Addr	Value
1001	a 10

▲ 1001번지에 숫자 10 저장

포인터

```
int *p = &a;
```

Addr	Value
2005	p 1001

▲ a의 주소(1001)를 저장

선언이 곧 사용법이다!

```
int *p;
```

"나중에 `*p`를 하면 `int`가 나온다"

어렵게 생각 말고,
"별(*)을 붙이면 정수가 되는 변수"라고 읽으세요.

핵심 연산자 2가지

&

주소 연산자 (Ampersand)

"너 몇 번째 줄에 있어?"

```
&a
```

결과: **1001** (행 번호)

*

참조 연산자 (Asterisk)

"그 줄로 이동해!"

```
*p
```

결과: **10** (그곳의 값)

코드와 엑셀의 연결

```
int a = 10; int *p = &a; // 포인터로 값 바꾸기 *p = 20;
```

Addr	Value
1001	a ~~10~~ → **20**
...	...
2005	p **1001**

🖱 ***p = 20;**의 의미:
p에 적힌 **1001번 줄로 이동(*)**해서
값을 **20**으로 바꿔라!

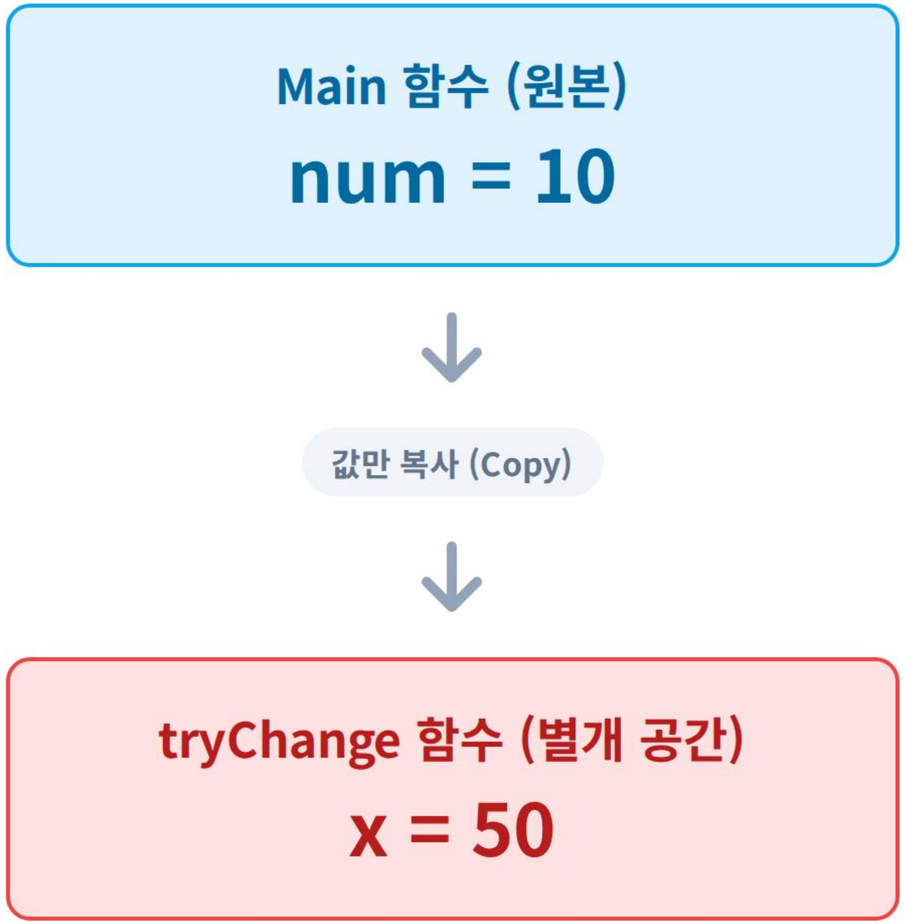
함수의 한계: Call by Value

함수에 변수를 그냥 넘겨주면,
"값(Value)의 복사본"만 전달됩니다.

함수 안에서 아무리 값을 바꿔도
원본(Main) 엑셀 시트에는 전혀 영향이 없습니다!

실패 사례: 값만 전달할 때

```
void tryChange(int x) { x = 50; // x만 바
꿈! } int main() { int num = 10;
tryChange(num); printf("%d", num); // 10 }
```



Call by Reference

~~값 복사~~

내용물만 전달

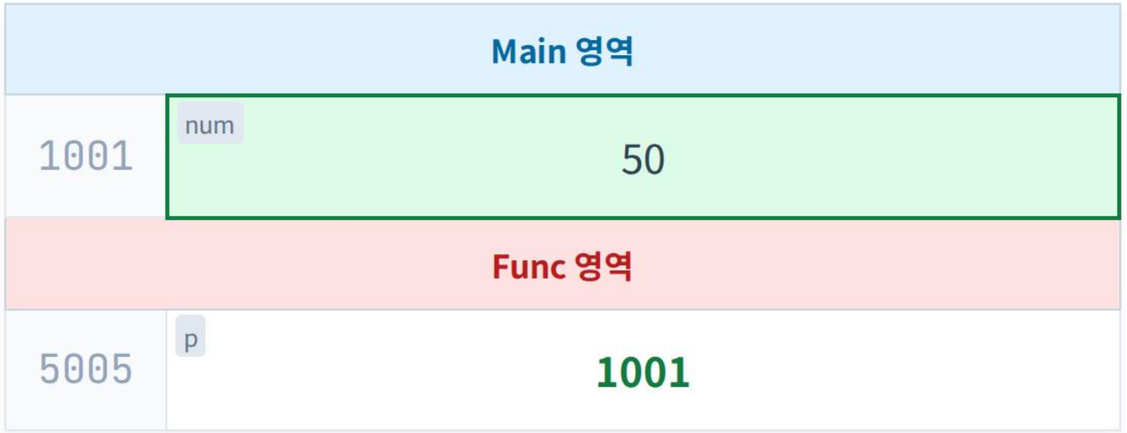
주소(&) 전달

🔑 **"내 데이터는**
1001번 줄에 있어!"

이제 함수가 **포인터(*)**를 이용해
직접 찾아와서 값을 바꿀 수 있습니다.

성공 사례: 포인터로 전달

```
void realChange(int *p) { *p = 50; // 그 주소로 가서 변경! } int main() { int num = 10; realChange(&num); // 주소 전달 printf("%d", num); // 50 }
```



✅ p가 1001번을 가리키므로
원격 제어 성공!

오늘의 요약

엑셀 시트

메모리는 주소(행 번호)가 있는
거대한 표다.

int *p

"나중에 *를 붙이면
int가 튀어나온다"고 읽자.

함수 전달

원본을 바꾸려면
값 말고 **주소(&)**를 보내라.

Q & A

포인터, 이제 엑셀처럼 보이시나요?

궁금한 점이 있다면 언제든 질문해주세요.