Item 8) Prefer nullptr to 0 and NULL

0과 NULL에 대한 고찰

0
정수 0은 포인터가 아니라 int이다. 일반적으로 null pointer를 사용해야 하는 경우에 0을 많이 사용하지만 c++의 주요 정책은 0은 pointer가 아니라 int이다.

NULL
C++에서 NULL은 타입의 불확실성을 가지고 있다. NULL은 보통 int로 정의되지만, 일부 구현에서는 long과 같은 다른 정수형으로도 정의될 수 있다. (표준에서 허용됨)

그러나 NULL의 정확한 타입은 중요하지 않다. 왜냐하면 0이나 NULL은 모두 포인터 타입이 아니라 정수형 타입이기 때문이다.

NULL또한 포인터를 나타내기 위해 자주 사용지만, 그 자체는 정수의 값이며 포인터 타입이 아니다. 따라서 NULL을 포인터처럼 사용하려고 하는 경오 타입 불일치가 발생한다.

```
void f(int);
void f(bool);
void f(void*);

f(0); // calls f(int), not f(void*)
f(NULL) // might not compile, but typically calls f(int). Never calls f(void*)
```

위의 예시를 한 번 살펴보자. 위의 경우에는 NULL을 argument로 가진다면 `f(int)`를 호출할 지, `f(bool)`을 호출할 지는 명확하지 않기 때문이다. (NULL이 float로 정의되어 있는 경우 어떤 변수로 타입 전환이 될 지 모르기 때문)

이런 이유 때문에, 0과 NULL을 사용하는 것이 아니라 `nullptr`를 사용하는 것이 더 좋은 선택이다.

그렇다면 `nullptr`는 뭔가요???

1. `nullptr`의 장점은 특정한 포인터 타입을 가지고 있지 않다는 것이다. `nullptr`는 `std::nullptr_t`이다. `std::nullptr_t`는 `nullptr`의 타입을 정의하는 타입이고. 순환적인 정의를 가지고 있다.

   즉 `std::nullptr_t`는 모든 `raw pointer type`으로 암시적으로 변환할 수 있다. 따라서 `nullptr`는 어떤 포인터 타입으로도 변환될 수 있어 "모든 타입의 포인터"처럼 작동된다.
2. `null_ptr`의 다른 장점은 cod의 명확성이 증가한다는 것이다.

```
auto result = findRecord(/* arguments */);

if (result == 0) {}
if (result == nullptr) {}
```

위의 예시를 보면 코드의 가독성 측면에서 더 나은 결과를 낳을 수 있다.