**항목 16. new 및 delete를 사용할 때는 형태를 반드시 맞추자.**

**이슈.**

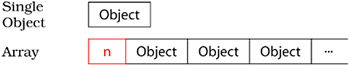
다음 코드의 잘못된 점이 있을까요?

|  |
| --- |
| std::string \*stringArray = new std::string[100]; ... delete stringArray; |

잘 되어 있는 것처럼 보입니다. stringArray의 new와 delete는 잘 매칭 되어 있는 것처럼 보입니다. 하지만 위 코드는 큰 문제점이 있습니다. 이 코드의 동작은 미정의 행동을 수행하게 되어 있습니다. stringArray의 스트링 객체 대부분의 소멸자가 제대로 호출 되지 않기 때문입니다.

사용자가 new 표현식을 사용할때는 두가지 일이 발생합니다. (1)메모리가 할당 되며, (2) 할당된 메모리에 하나 또는 그 이상의 생성자가 호출 됩니다. 사용자가 delete 표현식을 사용할 때 도 두가지 일이 발생합니다. (1) 하나 또는 그 이상의 소멸자들이 호출 되며, (2) 이후 메모리가 할당이 해제 됩니다. delete 표현식에서의 한가지 의문은 “메모리에서 얼마나 많은 객체들이 삭제가 될 것인가” 입니다. 이 의문의 답은 얼마나 많은 소멸자가 호출될 것인가 하는 데에 있습니다.

실제로, 이 의문은 단순합니다: 삭제되는 포인터가 단일 객체를 가리키는가 또는 객체들의 배열을 가리키는가? 이는 단일 객체들에 대한 메모리 레이아웃과 배열을 위한 메모리 레이아웃이 서로 다르기 때문에 매우 중요한 의문입니다. 특히, 배열에 대한 메모리는 배열의 크기를 대개 포함하며, delete가 얼마나 많은 소멸자들을 호출해야 하는지를 알기 쉽게 합니다. 단일 객체에 대한 메모리는 이 정보가 결여되어 있습니다. 이 레이아웃을 도시하면 다음과 같습니다. 여기서 n은 배열의 크기입니다:



한 포인터에 delete를 사용할 때, delete가 배열의 정보가 거기에 있는지에 대해 delete가 알 수 있는 유일한 방법은 사용자가 있다고 명시하는 것이다. 만약 delete에 [] 문구가 사용되면 delete는 배열이라는 것을 가정할 수 있습니다.

|  |
| --- |
| std::string \*stringPtr1 = new std::string; std::string \*stringPtr2 = new std::string[100]; ... **delete** stringPtr1;                       // delete an object **delete []**stringPtr2;                    // delete an array of objects |

규칙은 간단합니다. **new 표현식에 []을 사용한다면 delete 표현식에도 []를 사용**하면 되고, 만약 **new 표현식에 []를 사용하지 않는다면, 대응되는 delete 표현식에도 []를 사용하지 않으면 됩니다.**

이 규칙은 typedef 사용시에도 주의 해야 합니다. typedef 타입의 객체를 만들기 위해 new가 사용될 때 어떤 형태의 delete가 사용되어야만 되는지를 주의해야 한다는 것을 의미합니다.

|  |
| --- |
| typedef std::string AddressLines[4];   // a person's address has 4 lines, each of which is a string |

AddressLines는 배열이기 때문에, 아래의 new는 배열 형태의 delete와 매칭 되어야 합니다.

|  |
| --- |
| std::string \*pal = new AddressLines;   // note that "new AddressLines" returns a string\*,                                                   // just like "new string[4]" would  **delete pal;                           // undefined!** **delete [] pal;                        // fine** |

이러한 혼동을 피하기 위해, 배열 타입들에 대해서는 typedef들을 제한합니다. 또는, 표준 C++ 라이브러리는 string, vector가 존재 하므로, 이러한 템플릿들은 동적으로 할당된 배열들은 거의 안전하게 사용할 수 있는 방법들이 존재 합니다. 예를 들어 AddressLines는 string들의 vector로 정의가 가능 합니다. (vector<string> 타입)

**요약.**

\*new 표현식에 []를 썻으면, 대응되는 delete 표현식에도 []를 써야합니다. 마찬가지로 new 표현식에 []를 안썻으면, 대응되는 delete 표현식에도 []를 쓰지 말아야 합니다.