Item 15) Use constexpr whenever possible.

`constexpr`의 개념

1. *상수 표현* : `constexpr`는 `const`의 확장된 형태로, 값이 컴파일 타임에 결정된다고 선언할 때 사용된다. 이는 상수 값을 정의할 때 사용되며, 이 상수는 프로그램이 실행되기 전에 결정된다.
2. *함수 정의* : `constexpr` 함수는 컴파일 타임에 결과를 계산할 수 있는 함수를 정의하는 데 사용된다. `constexpr`함수는 컴파일 타임에 실행될 수 있으며, 이는 컴파일러가 이러한 함수를 최적화할 수 있는 기회를 재공한다.

왜 컴파일 타임에 알려지는 것이 좋은가요??

- 컴파일 타임에 알려진 값들은 `read-only memory`에 담겨진다. 이는 특히 임베디드 개발자에게 중요한 의미를 가진다.

```
int sz; // non-constexpre variable

constexpr auto arraySize1 = sz; // error sz's value not know at compilation

std::array<int,sz> data1; // error! same problem

constexpr auto arraySize2 = 10; // fine, 10 is an compile-time constatn

std::array<int, arraySize2> data2; // fine, arraySize2 is constexpr
```

- 간단하게 말해서 모든 `constexpr`객체는 `const`이다. 그러나 그 역은 성립하지 않는다.
  - `constexpr`는 컴파일 타임에 상수 값을 제공함.
  - `const`는 값이 변경되지 않음을 보장하지만, 컴파일 타임 상수를 보장하지 않음.

`constexpr` in Function

- 함수에서 `constexpr`은 객체와 다른 의미를 가진다.
  - `constexpr`함수의 argument에 컴파일 타임에 정해지는 값이 들어온다면, 함수의 결과는 컴파일 타임에 정해진다. 만약 argument들 중 하나의 값이라도 컴파일 타임에 정해지지 않는다면, 함수의 결과값은 rumtime에 정해진다

```
constexpr int pow(int base, int exp) noexcept
{
        ...
}

constexpr auto numConds = 5;
std::array<int, pow(3,numConds)> result;
```

- 위의 함수는 컴파일 타임에 결과를 정하기 때문에 컴파일 시점에 result가 만들어진다.
- 만약 base나 exp중 단 하나라도, 컴파일 타임에 정해지지 않는다면, pow의 결과는 runtime에 정해진다.

```
auto base = readFromDB("base"); // get these values at runtime
auto exp = readFromDB("exponent"); // ditto

auto baseToEXP = pow(base, exp); // call pow function at runtime
```

- 위의 예시에는 pow의 argument가 runtime에 정해지는 경우가 있으니 runtime에 정해진다

---

constexpr 함수의 return value

- C++11에서는 `constexpr`함수는 하나의 return을 가지고 있어야 한다. 즉, `if-else statement`가 있으면 안된다.
- 이에 단항 조건문인 경우에는 `?:`룰 사용해서 많이 사용했다.

```
constexpr int pow(int base, int exp) noexcept
{
        return (exp == 0 ? 1 : base * pow(base, exp -1));
}
```

- 그러나 C++14에서는 이러한 제약조건이 없어져서 복수의 return statement를 사용할 수 있게 되었다.

```
constexpr int pow(int base, int exp) noexcept
{
        auto result = 1;
        for(int i =0; i < exp; ++i) return *= base;

        return result;
}
```

Class안에서의 `constexpr`

- class안에서 생성자와 같은 return값이 void가 아닌 함수인 경우에 constexpr로 선언할 수 있다(C++11)

```
class Point {
public:
    constexpr Point(double xVal = 0, double yVal = 0) noexcept
        : x(xVal), y(yVal)
    {}

    constexpr double xValue() const noexcept { return x; }
    constexpr double yValue() const noexcept { return y; }

    void setX(double newX) noexcept { x = newX; }
    void setY(double newY) noexcept { y = newY; }

private:
    double x, y;
};


constexpr Point p1(9.4, 27.7); //fine
constexpr Point p2(28.8, 5.3); // also fine
```

- 가능한 한 `constexpr`를 많이 사용하라. `constexpr` 객체와 함수는 비 `constexpr` 객체와 함수보다 더 넓은 범위의 컴텍스트에서 사용될 수 있다.