Day-22

Control Flow

switch

- switch 문은 표현식(int, char, 열거형 클래스, 열거형)을 평가하고 일치하는 대/소문자 값과 연관된 문을 실행함

```
char x = "";
switch (x) {
    case 'a': y = 1; break;
    default: return -1;
}
return y;
```

- switch 범위

```
int x = 1;
switch(1){
    case 0: int x;      // 실행되지 않음
    case 1: cout << x; // 정의되지 않음
    case 2: {int y;}   // 변수 선언 가능, 초기화는 불가능
//case 3: cout << y; // 컴파일 에러
```

- C++17부터는 [[fallthrough]] 속성 사용 가능
  - [[fallthrough]]를 쓰면 case에 break를 쓰기 않아도 컴파일러가 경고를 내지 않음

```
char x = ...;
switch (x) {
    case 'a': x++; [[fallthrough]]; // C++17 : avoid warni
    case 'b': return 0
```

```
    default: return -1;
}
```

초기화문을 사용한 Control Flow

- C++17은 이니셜라이저가 있는 if문을 도입함

```
if(int ret = x + y; ret < 10)
    cout << ret;
```

- C++17은 이니셜라이저가 포함된 swich문을 도입함

```
switch(auto i = f(); x){
    case 1: return i + x;
```

- C++20은 이니셜라이저가 있는 범위기반 for 루프문 도입

```
for(int i = 0; auto x : {'A', 'B', 'C'})
    cout << i++ << ":" << x << " "; // print: 1:A 2:B: 3:C
```

사용되지 않는 변수 경고 방지 : [[maybe_unused]]

- 대부분의 컴파일러는 변수가 사용되지 않을때 경고 표시
- C++17은 [[maybe_unused]] 속성을 도입하여 이 경고를 방지함

```
int f(int value){
    [[maybe_unused]] int x = value;
#if defined(ENABLE_SQUARE_PATH)
    return x * x;
#else
// static_cast<void>(x); // C++17 이전
    return 0;
#endif
}
```

```
template<typename T>
inf f([[maybe_unused]] T value){
    if constexpr (sizeof(value) >= 4)
        return 1;
    else
        return 2;
}

template<typename T>
int g([[maybe_unused]] T value){
    using R = decltype(value);
    return R{};
}

// [[maybe_unused]]가 적용되지 않은 경우 MSVC가 경고 표시
```