# 2023 08 02

**Reference Collapsing**

```
/*
Reference Collapsing
T&    &     ===> T&
T&    &&    ===> T&
T&&   &     ===> T&
T&&   &&    ===> T&&

L value ref --> int &x
R value ref --> int &&x
Universal ref  --> auto &&x
*/
```

**Universal Reference**

Bir universal reference her değer kategorisindeki ifadeye bağlanabilir.

- L value ya da R value (PR || X)
- const ya da non const

auto &&r x;

Eğer ilk değer veren ifadenin (x) değer kategorisi L value ise o zaman auto için yapılan çıkarım L taraf referans türü olur ve r değişkenin türü "reference collapsing" ile T &

Eğer ilk değer veren ifadenin (x) değer kategorisi R value ise o zaman auto için yapılan çıkarım referans olmayan türü olur ve r değişkenin türü ise T && olur.

```
int main()
{
    int x = 5;

    auto &&r1 = x;
    auto &&r2 = 10;

    const int y = 67;
    auto &&r3 = y;
}
```

```
int main()
{
    auto &&x = 10; // int && x = 10

    int ival {4};
    auto&& x = ival; // int& && --> int& x = ival
}
```

**Decltype Specifier**

```
    decltype(expr) --> bir tür döner
    compiler time ile ilgili

    decltype specifier ile yapılan tür çıkarımınında iki ayrı
    kural seti vardır.

    - 1) aldığı üyenin isim olması
        decltype(x)
        decltype(ptr->x)
        decltype(a.b)
    - 2) aldığı üyenin isim olmaması
        decltype(10)
        decltype(x + 5)
        decltype((x))
```

```
// decltype 1.kural
int main()
{
    int x = 120;
    decltype(x); // int

    const int y = 21;
    decltype(y); // const int

    decltype(y) z = 10;
}
```

```
int main()
{
    int x = 5;
    int& r{ x };

    decltype(r) y = x;
}
```

```
int main()
{
    int&& r = 10;
    // && & --> &x = 56 hata
    decltype(r)& x = 56; // syntax hatası
    // && && -> y = 32;
    decltype(r)&& y = 32; // geçerli
}
```

```
int main()
{
    int a[5]{};
    // int[5] --> int b[5] = a
    decltype(a)b = a; // hatalı
    decltype(a)b = { 1, 2, 3}; // hata yok
}
```

```cpp
int main()
{
    int x = 56;
    decltype(x)* p = &x; // int *p = &x
}
```

```cpp
int main()
{
    int a[20] {};
    // int (*p)[20] = &a;
    decltype(a)* p = &a;
}
```

```cpp
// decltype 2.Kural
/*
    Diyelim ki T bir tür olmak üzere expression ifadesinin türü
    T olsun.
    Eğer expression ifadesinin primary value kategorisi:
        PR value ise elde edilen türü T
        L value ise elde edilen türü T&
        X value ise elde edilen türü T&&

*/
```

```cpp
int main()
{
    int x = 10;
    // x + 5 --> PR Value
    decltype(x + 5); // --> int
}
```

```cpp
int main()
{
    int a[5]{};
    // a[2] --> L value
    decltype(a[2]);  // --> int&
}
```

```cpp
int main()
{
    int x{ 435 };
    int *p{ &x };
    // *p --> L value
    decltype(*p); // --> int&
}
```

```cpp
int foo(); // foo() --> PR value
int& bar(); // bar() --> L value
int&& baz(); /// baz() --> X value

int main()
{
    decltype(foo()); // --> int
    decltype(bar()); // --> int&
    decltype(baz()); // --> int&&
}
```

```cpp
int main()
{
    int x = 10;
    int y = 20;
    // 1. Kural seti
    decltype(x) a = y; // int a = y;
    // 2. Kural seti
    decltype((x)) b = y; // int& b = y;
}
```

```cpp
int main()
{
    const char* p[] = { "eren", "furkan", "melike" };
    using nectype = decltype(p); // const char *[3];

    nectype x;
}
```

**Unevaluated Context (işlem kodu üretilmeyen bağlam)**

- sizeof
- decltype
- typeid
- noexcept

```cpp
int main()
{
    int x = 12;
    auto val = sizeof(++x); // val = 12
}
```

```cpp
int main()
{
    // decltype -> unevaluated context
    int x = 10;
    // ++x --> L value
    decltype(++x) y = x; // int& y = x

    cout << "x = " << x << "\n"; // x = 10
    ++y;
    cout << "x = " << x << "\n"; // x = 11
}
```

```
// variadic fonksiyon
void foo(int, ...);

int main()
{
    // ... olan yere istediğimiz kadar arguman gönderebiliriz.
    //foo(3, x, y z);
}
```

**Default Arguman**

```
// default arguman
int func(int = 1; int = 2; int 3);
int main()
{
    func(50, 60, 70);
    func(50, 60);
    func(50);
    func();
}
```

```
int y = 10;
void func(int = ++y)
{

}
int main()
{
    func();
    func();
    func();
    cout << "y = " << y << "\n"; // y = 13
}
```

```
void foo(const char *p = "emre"); // geçerli
void foo(const char*= "emre"); // geçersiz
void foo(const char* = "emre"); // geçerli
```

```
int main()
{
    using namespace std;

    int a = 10;
    int b = 40;
    // ilk token en uzun olacak şekilde
    int c = a+++b; // a++ b;

    cout << "a = " << a << "\n"; // 11
    cout << "b = " << b << "\n"; // 10
    cout << "c = " << c << "\n"; // 50
}
```

```
//
void foo(int x, int y = x); // syntax hatası
```

**constexpr (C++11)**

```
int main()
{
    const int y = 45;
    const int x = y;

    int b[y] = {0}; // geçerli
    int a[x] = {0}; // geçersiz
}
```

```
int main()
{
    // bir sabit ifadesi
    constexpr int x = 10; // x'in türü const int

    int y = 43;
    constexpr int z = y; // illegal
}
```