2023 07 31

Reference Semantiği

```
int main()
{
    int x = 10;
    const int *p = &x; // legal

    const int y = 10;
    int *p1 = &y; // legal değil
}
```

```
// YAPILMAMASI GEREK
int *func(void)
{
    // dangling pointer olur
    int x = 10; // otomatik ömürlü
    return &x;
}
```

```
int *func(void)
{
    // legal
    static int x = 10;
    return &x;
}
```

```
//trailing return type
auto func(void) -> int (*)(int)
{

}
```

```
struct Data {
  int x, y, z;
}

Data& bar(Data &r)
{
    // aldığı nesneyi döndürür
  return r;
}

int main()
{
  Data mydata{ 1, 4, 2};
  Data &dr = bar(mydata);
}
```

```
// Eğer referans const L value referans ise R value expression ile ilk değer
verebiliriz
int main()
{
    int x = 10;
    double &r = x; // illegal -- tür uyumsuzluğu

    const double &r1 = x // legal
    // aslında derleyicinin yaptığı bu
    double temp_obj = ival;
    const double& fr = temp_obj;

    const int& r2 = 10;  // legal

    int& r3 = 10; // illegal
    // aslında derleyicinin yaptığı bu
    int temp_object = 10;
    const int &r = temp_object;
}
```

```
/*
                L value expr     const L value expr   R value expr
    T &         bağlanabilir        bağlanmaz          bağlanmaz

    const T&    bağlanabilir        bağlanabilir       bağlanabilir
*/
```

```
                    Pointer vs Referans
    pointer semantiği               referans  semantiği
1)  default init edilir              default init edilemez
2)  pointer to pointer olr           referans to referans olmaz
3)  nullpointer var                  null referance yok
```

```
/*
    int &r      L value referance
    int &&r     R value referance
    auto &&r    Universal referance
*/
```

R value referance

```
int foo();
int main()
{
    int x = 20;
    // R value refler R value expresion bağlanabilir
    int && r = 10; // geçerli
    int && r1 = x; // geçersiz

    int &&r = foo(); // geçerli
}
```

Type Deduction (tür çıkarımı)

- auto
- decltype
- decltype(auto)

Auto Type Deduction

```
int main()
{
    // tür çıkarımı x için değil auto için yapılır
    auto x = 10;
    auto y; // default init  syntax hatası
}
```

```
int main()
{
    const int x = 10;
    // const'luk düşer
    auto y = x; // y nun türü int
}
```

```
int main()
{
    int x = 10;
    int &r = x;
    auto y = r; // y -> int
}
```

```
int main()
{
    int x = 10;
    const int &r = x;
    auto t = r; // t -> int
}
```

```
int main()
{
    int a[10]{};
    auto b = a; // b -> int *

    const x[10]{};
    auto y = x; // y -> const int *
}
```

```
int main()
{
    auto ps = "emre"; // auto -> const char*
}
```

```cpp
int func(int); // func'un türü int(int)
// &func ifadenin türü int(*)(int)

int main()
{
    auto x = func; // int(*x)(int)
    auto y = &func; // int(*y)(int)
}
```

```cpp
int main()
{
    int x = 10;
    int *p = &x;
    // top level const
    const auto y = p // y --> int *const y
}
```

```cpp
int main()
{
    char c1 = 10;
    char c2 = 20;

    auto x = c1; // x -> char
    auto y = +c1; // y -> int
    auto z = c1 + c2; // z -> int
}
```

```cpp
int main()
{
    const int x = 10;
    auto& y = x; // auto -> const int
    // y -> const int&
}
```

```cpp
int main()
{
    int a[5]{};
    auto &b = a; // auto -> int[5]
    // b -> int(&)[5]
}
```

```cpp
int main()
{
    auto &x = "eren"; // auto -> const char[5]
    // x -> const char(&)[5]
}
```

```cpp
int foo(int);
int main()
{
    auto &f = foo; // auto -> int(int)
    // f -> int(&)(int)  function referance
}
```

```
// using bildirimi

using Word = int;
using ciptr = const int*
using inta20 = int[20];
using FCMP = int(*)(const char*, const char*);
```

Reference Collapsing

```
/*
    T&      &    ==> T&
    T&      &&   ==> T&
    T&&     &    ==> T&
    T&&     &&   ==> T&&
*/
```