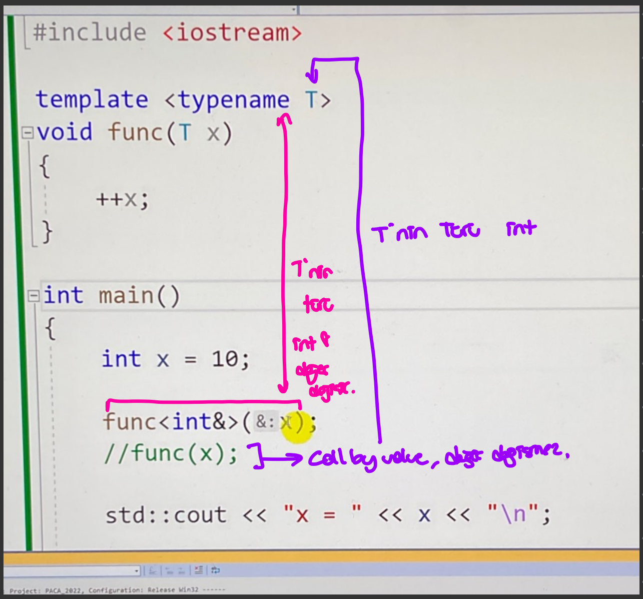
43'ı attadım, 40'ın başında tuple anlattı

* Std::Functional:

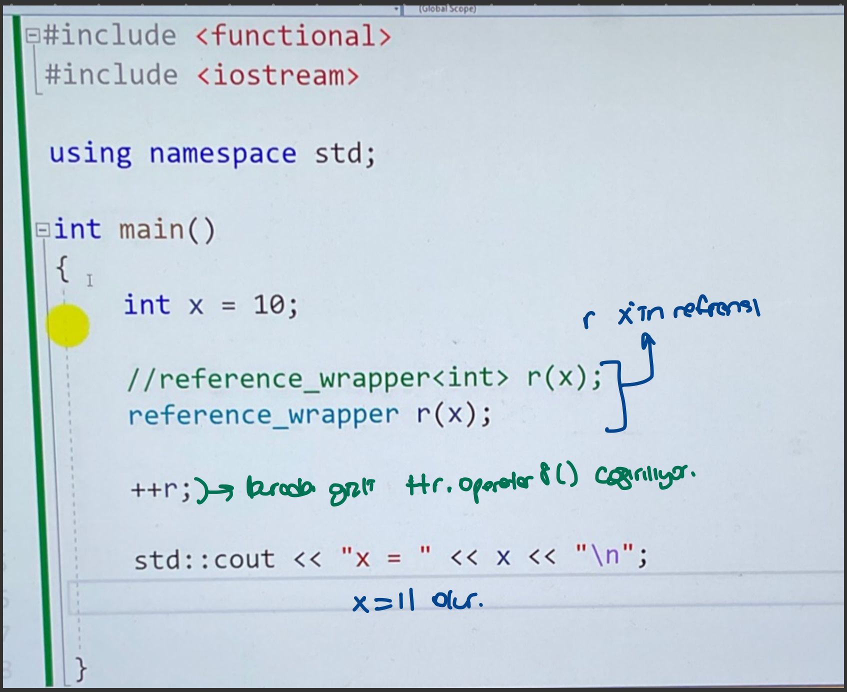
- Reference wrapper:

- Genel faydası, normal wrapperlar gibi, bize bir interface sağlayacak.
- Rebindable references
- Containerlarda referans wrapper saklanabilir.



```cpp
#include <iostream>

template <typename T>
void func(T x)
{
    ++x;
}

int main()
{
    int x = 10;

    func<int&>(x);
    //func(x);

    std::cout << "x = " << x << "\n";
```

T'nin türü int
T'nin türü int& değer değişir.
call by value, değer değişmez.



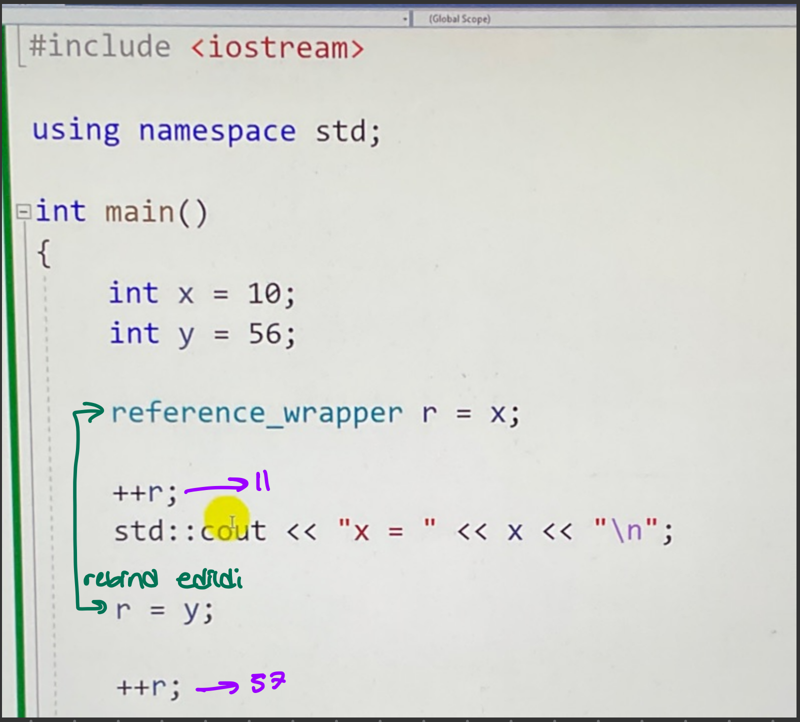
```cpp
#include <functional>
#include <iostream>

using namespace std;

int main()
{
    int x = 10;

    //reference_wrapper<int> r(x);
    reference_wrapper r(x);

    ++r;

    std::cout << "x = " << x << "\n";

}
```

→ operator &() bize örtülü dönüşüm imkanı verir.

↓

yani referanswrapper <int> = int yapabiliriz.

→ rebind example

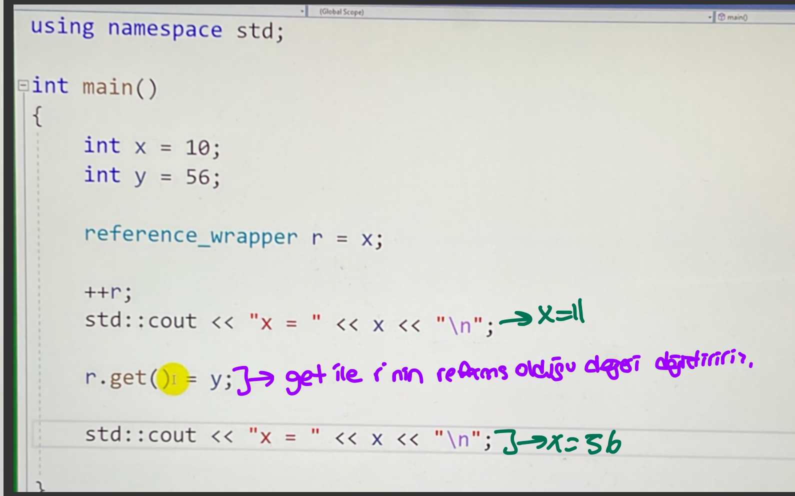


```cpp
#include <iostream>

using namespace std;

int main()
{
    int x = 10;
    int y = 56;

    reference_wrapper r = x;

    ++r;
    std::cout << "x = " << x << "\n";

    r = y;

    ++r;
```

++r; → 11

rebind edildi

++r; → 57



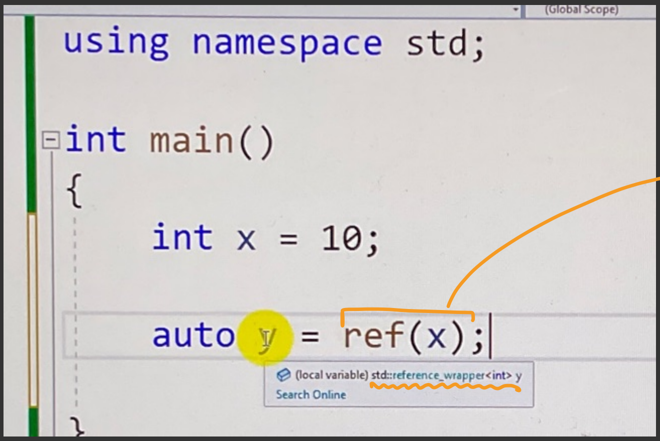
```cpp
using namespace std;

int main()
{
    int x = 10;
    int y = 56;

    reference_wrapper r = x;

    ++r;
    std::cout << "x = " << x << "\n";

    r.get() = y;

    std::cout << "x = " << x << "\n";
}
```

→ x=11

→ get ile r'nin referans olduğu değeri değiştiririz.

→ x=56

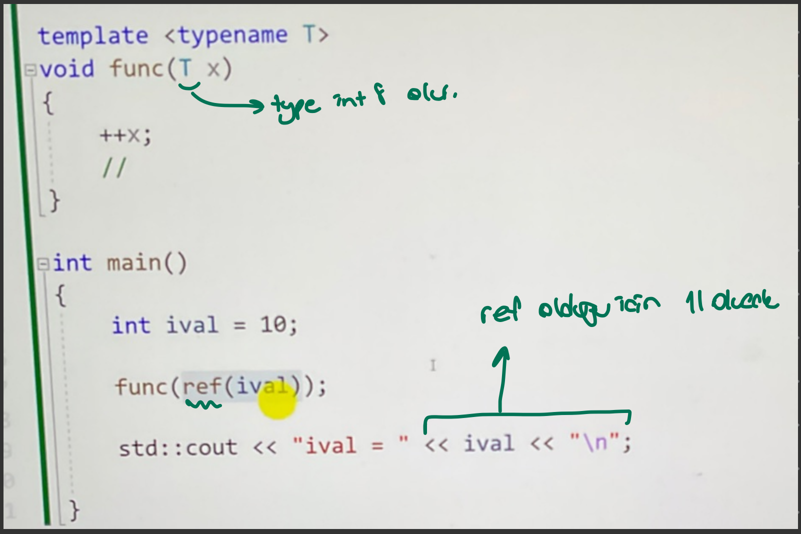
↳ get'in return değeri T&



```cpp
using namespace std;

int main()
{
    int x = 10;

    auto y = ref(x);
}
```

(local variable) std::reference_wrapper<int> y
Search Online

→ Ref() adında bir template fonksiyonu var

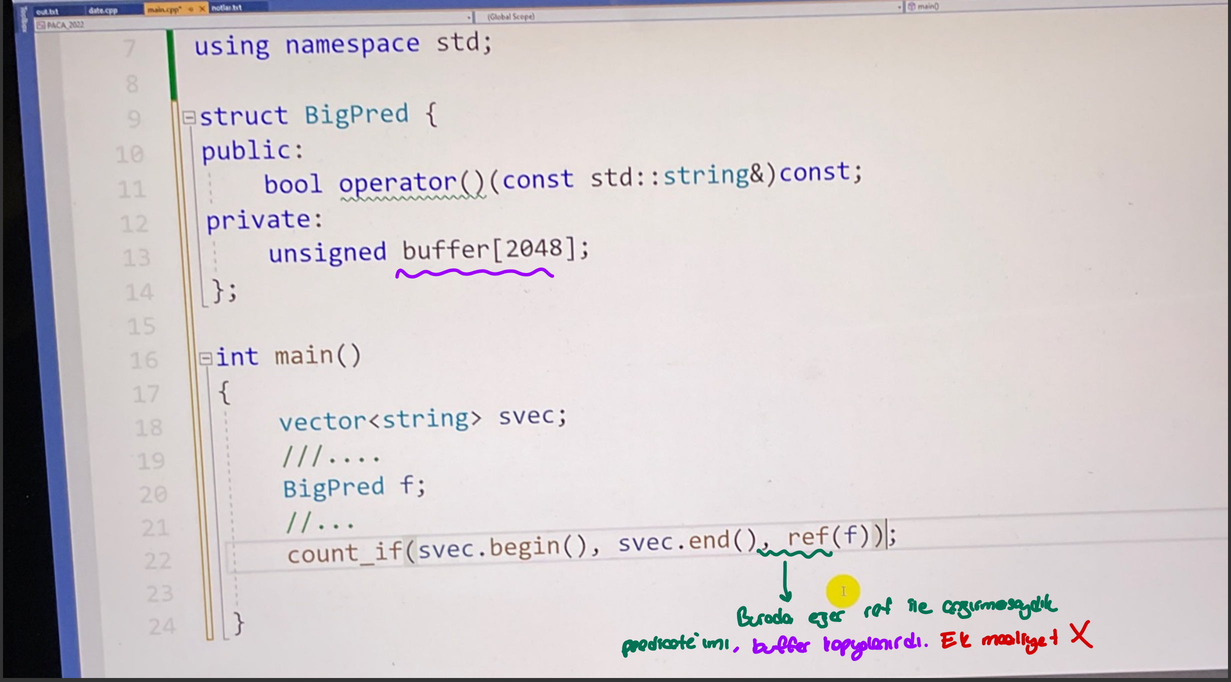


```cpp
template <typename T>
void func(T x)
{
    ++x;
    //
}

int main()
{
    int ival = 10;

    func(ref(ival));

    std::cout << "ival = " << ival << "\n";
}
```

→ type int & olur.

ref olduğu için 11 olacak

→ Ayrıca, cref de var const açılım verir.



```cpp
using namespace std;

struct BigPred {
public:
	bool operator()(const std::string&)const;
private:
	unsigned buffer[2048];
};

int main()
{
	vector<string> svec;
	///....
	BigPred f;
	//...
	count_if(svec.begin(), svec.end(), ref(f));

}
```

Burada eğer ref ile çağırmasaydık predicate'imi, buffer kopyalanırdı. Ek maliyet X