

[Sign up](#)[Code](#)[Pull requests](#) 0[Projects](#) 0[Actions](#)[Security](#)[Pulse](#)

Branch: master ▾

[Find file](#)[Copy path](#)[Cplusplus_Ders_Notlari](#) / [implicit_this_capture.md](#)

Fetching contributors...



93 lines (78 sloc) | 2.15 KB

[Raw](#)[Blame](#)[History](#)

C++20 standartları `this` göstericisinin örtülü (`implicit`) olarak yakalanmasını "deprecated" olarak belirliyor. C++23 standartları bu özelliği dilden tamamen kaldırılabileceğinden kullanmamakta fayda var. Aşağıdaki koda bakalım:

```
#include <iostream>

struct Nec {
    int mx = 5;
    //
    void func()
    {
        auto f = [=] {std::cout << mx; };
        f();
        //
    }
};

int main()
{
    Nec{}.func();
}
```

`Nec` sınıfının `func` üye işlevi içinde kullanılan `lambda` ifadesinde sınıfın `mx` isimli veri ögesinin kullanıldığını görüyorsunuz. Burada yakalanan (`capture edilen`) edilen `mx` veri ögesi değil `this` göstericisi. Yani `mx` kapanış (`closure`) sınıfının bir veri ögesine kopyalanmıyor. Kod içinde kullanılan `mx`, veri ögesinin ta kendisi. (`this->mx`). Aşağıdaki kod bunu göstermeye yönelik:

```
#include <iostream>

struct Nec {
    int mx = 5;
    //
    void func()
    {
        auto f = [=] {++mx; };
        f();
        //
    }
};

int main()
{
    Nec nec{};

    nec.func();
    std::cout << nec.mx << '\n';
}
```

Kodu çalıştırdığınızda `nec` isimli nesnenin `mx` ögesinin değerinin değiştiğini göreceksiniz. `this` göstericisinin bu şekilde örtülü olarak yakalanması bir kopyalama söz konusu olduğunu düşünen programcılar zor duruma düşürebiliyor. Aşağıdaki koda bakalım:

```
#include <functional>

struct Nec {
    int mx;
    auto get_closure()
    {
        return [=] { return mx; };
    }
};

int main()
{
    std::function<int()> f;
    {
        Nec nec{10};
        f = nec.get_closure();
        // nec nesnesinin hayatı burada sonlanıyor
    }
    int result = f(); //tanımsız davranış
}
```

C++17 standartları ise `this` göstericisinin referansla yakalanmasının yanında hem de kopyalama yoluyla isimle (`explicit`) yakalanmasını mümkün kıldı:

```
struct Nec {
    int mx = 5;
    void f1()
    {
        auto f1 = [this] {++mx; };
        //
        auto f2 = [*this]{/* */ };
    }
};
```

`this` göstericisinin örtülü olarak referans semantiği ile yakalanmasında ise C++20 standartları bir değişiklik

getirmedi.

Kaynak : C++20 Deprecate implicit lambda capture of this

© 2020 GitHub, Inc.

[Terms](#)

[Privacy](#)

[Security](#)

[Status](#)

[Help](#)

[Contact GitHub](#)

[Pricing](#)

[API](#)

[Training](#)

[Blog](#)

[About](#)