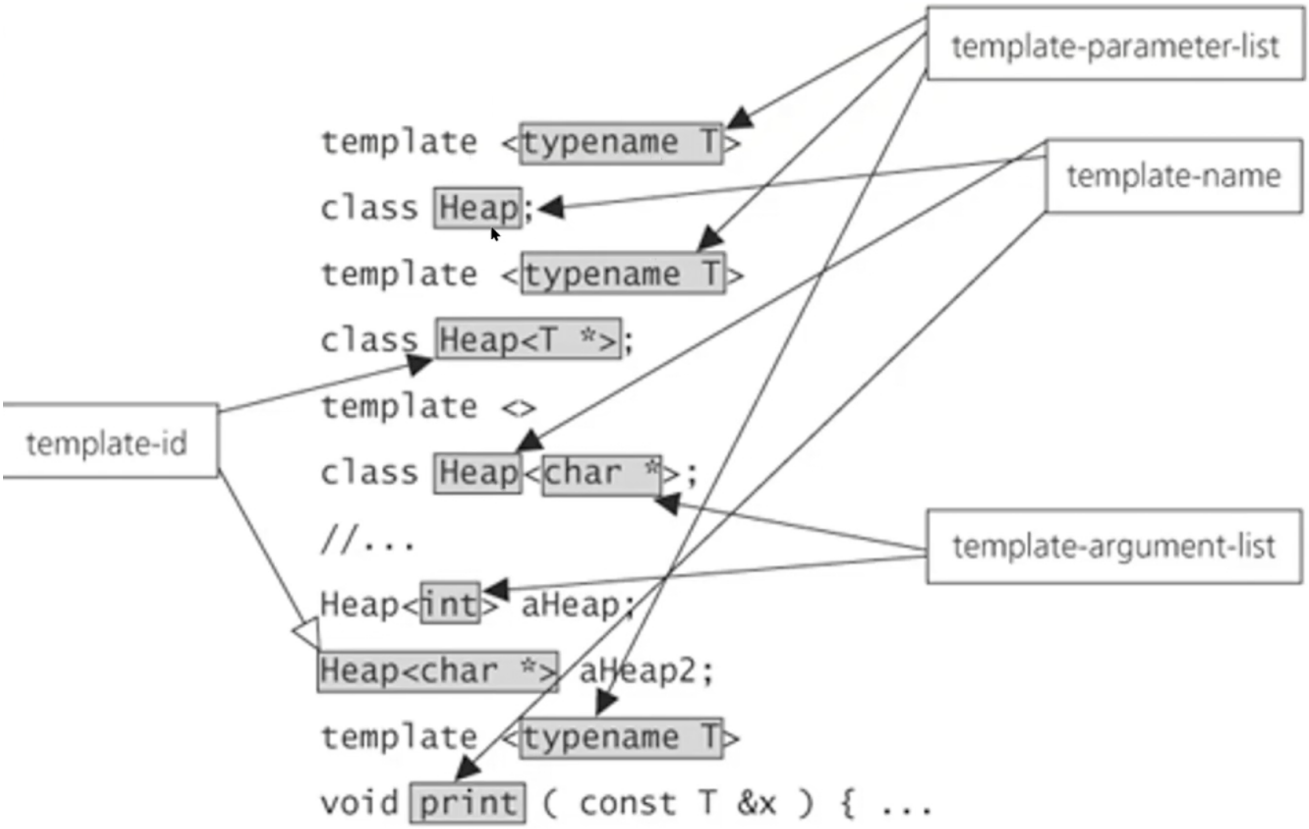
15.hafta

İçindekiler

26_30_09_2023

Template Temel Hızlı Tekrar



Template argument deduction: template argümanlarını bilme aşamasında, argümanların türünün bilinmesine deniyor

En çok yapılan deduction hataları

```
template <typename T>
void foo(T);

template <typename T>
```

```
void foo(T&);

template <typename T>
void foo(T&&);
```

- 1.için referanslık ve constluk düşüyor.
- 2.için constluk düşmüyor.

Çıkarım T'için yapılıyor, parametre için değil.

Diziler için sol taraf referansı olduğunda çıkarım int[] olarak yapılıyor. Array decay olmuyor.

bazı durumlarda çıkarımın yapılması kodun çağırılıp çağırılmasına bağlı değil

```
template <typename T>
void foo(T,T);

int main()
{
    void (*fp)(int,int) = &foo;
}
```

- Çıkarım için incomplete type türünden bir sınıf tamamlayın ve bu şekilde çıkarımın nasıl yapıldığını görebileceğiz

```
template<typename T>
class TypeTeller;
```

```
template<typename T,typename U>

void foo(std::array<T,,sizeof(U)>,std::array<U,sizeof(T)>);

int mian()
{
    std::array<int,sizeof(double)> a;
    std::array<double,sizeof(int)> b;
    std::array<double, 5> c;
    foo(a,b);
    //foo(a,c); // Sentaks hatası oluşur çünkü, 1. parametre int oldu,
sizeof(double) ve 5 aynı değil.
}
```

Çıkarım konusunda cpp ref örnek

```
1
2
3   template<typename T>
4   void f1(T*);
5
6   template<typename E, int N>
7   void f2(E(&)[N]);
8
9   template<typename T1, typename T2, typename T3>
10  void f3(T1(T2::*)(T3*));
11
12  class S {
13  public:
14      void f(double*);
15  };
16
17  void g(int*** ppp)
18  {
19      bool b[42];
20      f1(ppp); // deduces T to be int**
21      f2(b); // deduces E to be bool and N to be 42
22      f3(&S::f); // deduces T1 = void, T2 = S, and T3 = double
23  }
```

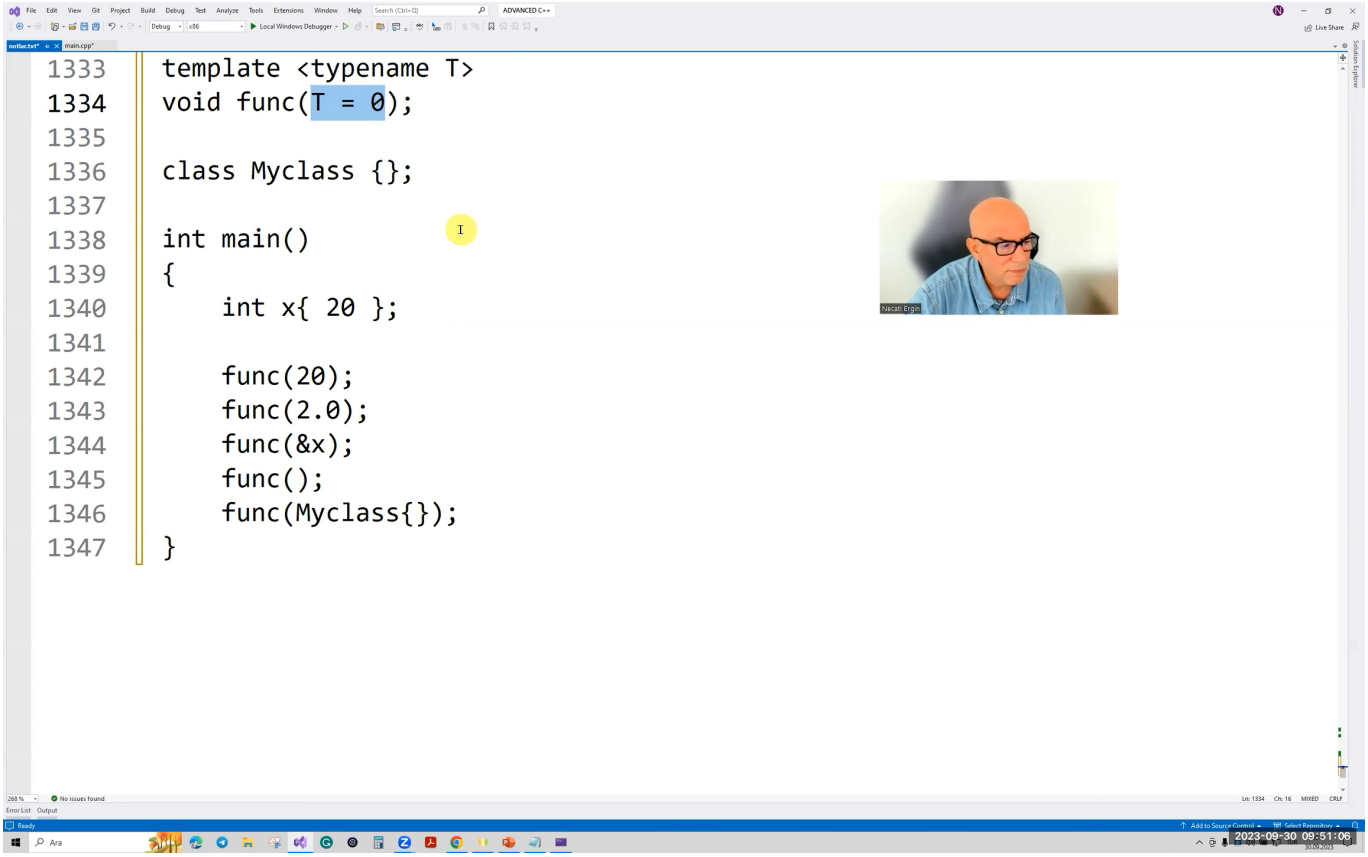
Fonksiyonun parametresi fucntion pointer türünden template olabilir.

```
template <typename T, typename U>
void func(T(*(U)));
```

- Pozitif lmabda idiomu

```
#include <vector>
template‹typename T>
void func(T&& x, const std::vector‹T>& ivec);
int main()
{
    using namespace std;
    vector‹int › ivec(10);
    func(ivec[0], ivec);
}
```

- template parametresi default argüman alabilir. Varsayılan argümandan hareketle çıkarım yapılamaz. Myclass için sentaks hatası olmaz.



Non-Type Template Parametreler

Tam sayı türlerinden olabilir. Objec türlerden olabilir. Referans türü olabilir. Fonksiyon pointer'ı ve member fonksiyon türü olabilir. Gerçek sayı türleri de olabilir.

```
template <int x>
class Nec{};
template <int*>
class Den{};
```

```
class Myclass
{ public: double foo(double); };
int g{};
int foo(int);
template<auto x>
class A{};
template <int (*pf)(int)>
class C{};
template<double(Myclass::*)(double)>
class D{};
template<int &>
class E{};

int main()
{
    int ival{};
    A<5>ax;
```

```
    B<&g> bx;
    C<foo> cx;
    D<Myclass::foo> dx;
    E<g>ex;
}
```