**2023.10.20**

**Template**

**template** **<typename** T**>**

**class** Myclass **{};**

**template** **<typename** T**>**

**class** Nec**{};**

**int** main**()**

**{**

***// Myclass'ın int spec Nec'in Myclass<int> spec'inde kullanılır***

Nec**<**Myclass**<int>>;**

**}**

**template** **<typename** T**>**

**class** Myclass **{};**

**int** main**()**

**{**

***// farklı türlerde nesneler***

Myclass**<double>** dm**;**

Myclass**<int>** im**;**

***// dm = im; böyle bir dönüşüm yok***

**}**

**template** **<int>**

**class** Myclass

**{};**

**int** main**()**

**{**

***// farklı türlerde nesneler***

Myclass**<**5**>** dm**;**

Myclass**<**6**>** im**;**

***// dm = im; böyle bir dönüşüm yok***

**}**

**template** **<typename** T**>**

**class** Myclass

**{**

**public:**

**void** foo**(**T x**)**

**{**

**++**x**;**

**}**

**};**

**class** Nec**{};**

**int** main**()**

**{**

***// burda syntax hatası olmaz çünkü foo'yu daha yazmadı derleyici***

Myclass**<**Nec**>** m**;**

***// burda syntax hatası olur derleyici foo'yu yazdı ve ++x diye bir şey olmaz hata***

Nec mynec**;**

m**.**foo**(**mynec**);**

**}**

**Template Fonkisyonlarını Tanımlama**

Cpp dosyasına yazamayız çünkü derleyici kodu görmeli.

***// myclass.h***

**template<typename** T**>**

**class** Myclass

**{**

**public:**

T bar**(**T**);**

***// burada inline olarak tanımlanabilir***

**void** foo**(**T x**)**

**{**

x**.**f**();**

**++**x**;**

x **=** 0**;**

**}**

**};**

T bar(T) burada header dosyasında tanımlanabilir.

***// myclass.hpp***

#include "myclass.h"

***// burada da tanımlayabiliriz***

**template** **<typename** T**>**

T Myclass**<**T**>::**bar**(**T x**)**

**{**

***// code***

**}**

***// myclass.h***

**template<class** T**,** **class** U**>**

**class** Myclass

**{**

**public:**

**void** foo**(int x);**

**void** bar**(**T**,** T**)**

**};**

**template<typename** T**,** **typename** U**>**

**void** Myclass**<**T**,** U**>::**foo**(int** x**)**

**{**

**}**

**template<typename** T**>**

**void** Myclass**<**T**,**U**>::**bar**(**T**,** T**)**

**{**

**}**

**template** **<typename** T>

**class** Myclass **{};**

**template** **<typename** T**>**

**void** func**(const** Myclass**<**T**>&);** ***// T'in çıkarımı int oldu***

**int** main**()**

**{**

Myclass**<int>** x**;**

func**(**x**);**

**}**

**template** **<typename** T**>**

**class** Myclass**{};**

**bool** **operator==(const** Myclass**<int>&,** **const** Myclass**<int>&)**

**{**

***// Myclass'in int speclerini karşılaştırabiliriz;***

**}**

**template** **<typename** T**>**

**bool** **operator==(const** Myclass**<**T**>&,** **const** Myclass**<**T**>&)**

**{**

***// Myclass bütün spec'lerinin karşılaştırabiliriz***

**return** **true;**

**}**

***////////////***

**template** **<typename** T**>**

**class** Myclass

**{**

**public:**

**void** foo**(**T**&&);**

**};**

**int** main**()**

**{**

Myclass**<int>** m**;**

m**.**foo**(**12**);** ***// universal ref değil sağ taraf ref***

**}**

**template** **<typename** T**>**

**class** Nec

**{**

**public:**

**void** myNec**()**

**{**

***// class scope'ta hiçbir farkı yok aşağıdaki ikisinin***

Nec nec

Nec**<**T**>** nec

**}**

**}**

**template<typename** T**>**

**class** Nec

**{**

**public:**

Nec func**(**Nec x**);**

**template<typename** U**>**

**void** bar**(**Nec**<**U**>** x**)**

**{**

**}**

**}**

**template** **<typename** T**>**

Nec**<**T**>** Nec**<**T**>::**func**(**Nec**<**T**>** x**)**

**{**

**return** x**;**

**}**

**int** main**()**

**{**

Nec**<double>** x**;**

Nec**<int>** y**;**

x**.**func**(**y**);** ***// error***

x**.**bar**(**y**)** ***// error yok***

**}**

**template<typename** T**>**

**class** Nec

**{**

**public:**

**template<typename** U**>**

**void** func**(**Nec**<**U**>** x**)**

**{**

std**::**cout **<<** **typeid(\*this).**name**()** **<<** "\n"**;** ***// Nec<int>***

std**::**cout **<<** **typeid(**x**).**name**()** **<<** "\n"**;** ***// Nec<double>***

**}**

**}**

**int** main**()**

**{**

Nec**<int>** x**;**

Nec**<double>** y**;**

x**.**func**(**y**);**

**}**

**template<typename** T**,** **typename** U**>**

**struct** Pair

**{**

Pair**()** **=** **default;**

Pair**(const** T**&** t**,** **const** U**&** u**)** **:** first**(**t**);** second**(**u**)**

**{**

**}**

T first**{};**

U second**{};**

**}**

**int** main**()**

**{**

**using** **namespace** std**;**

pair p1**{**3**,** 5.6**};** ***// CTAD***

**}**

**Untype Parametre**

**template** **<int** n**>**

**class** Myclass **{};**

untype parametre:

* tam sayi turleri (int, short)
* enum turleri
* object pointer / reference
* function pointer
* member function pointer

***// Cpp 17 ve eskileri için bu kod hatalı***

**template** **<double** n**>**

**class** Myclass **{};**

**template** **<int** **(\***fp**)(int)>**

**class** Myclass **{};**

**int** foo**(int)**

**int** main**()**

**{**

Myclass**<&**foo**>** m**;**

**}**

**template** **<int** x**>**

**class** Myclass **{};**

**template** **<long** x**>**

**class** Myclass **{};**

**int** main**()**

**{**

Myclass**<**5**>** m1**;** ***// sytnax hatası olur***

**}**

***// yukarıdaki yerine***

**template** **<typename** T**,** T x**>**

**class** Myclass **{};**

**int** main**()**

**{**

Myclass**<int,** 5**>** x**;**

Myclass**<long,** 5**>** y**;**

**}**

***// cpp 17***

**template** **<auto** n**>**

**class** Myclass **{};**

**int** main**()**

**{**

***// derleyici çıkarım yapar***

Myclass**<**5**>** m1**;**

Myclass**<**1u**>** m2**;**

Myclass**<**'A'**>** m3**;**

**}**

**Default Template Arguman**

***// default template arguman***

**template** **<typename** T **=** **int,** **typename** U **=** **long>**

**class** Myclass **{};**

**int** main**()**

**{**

Myclass**<>** m1**;**

std**::**cout **<<** **typeid(**m1**).**name**();** ***// Myclass<int,long>***

**}**

**template** **<int** x **=** 10**,** **int** y **=** 20**>**

**class** Myclass **{};**

**int** main**()**

**{**

Myclass**<**3**,** 5**>** m1**;**

Myclass**<**3**>** m2**;**

Myclass**<>** m3**;**

**}**

**template** **<typename** T**,** std**:size\_t** N**>**

**constexpr** **auto** asize**(const** T**(&)[**N**])**

**{**

**return** N**;**

**}**

**int** main**()**

**{**

**int** a**[]{**1 **,**2 **,**3**,** 4**};**

**constexpr** **auto** size **=** asize**(**a**);**

**}**

1. explicit specialization (full specialization)
2. partial specialization

**Explicit Specialization**

**template** **<typename** T**>**

**class** Nec

**{**

**public:**

Nec**()**

**{**

std**::**cout **<<** "primary template type T is : "

**<<** **typeid(**T**).**name**()** **<<** "\n"**;**

**}**

**};**

***// explicit specialization***

***/\****

***T 'ın int olduğu duurunlarda bu kullanılır***

***\*/***

**template<>**

**class** Nec**<int>**

**{**

**public:**

Nec**()**

**{**

std**::**cout **<<** "explicit template type T is : "

**<<** **typeid(**T**).**name**()** **<<** "\n"**;**

**}**

**};**

**template** **<typename** T**,** **typename** U**,** **int** N**>**

**class** Myclass**{};**

**template<>**

**class** Myclass**<int,** **double,** 20**>**

**{**

**public:**

Myclass**()**

**{**

std**::**cout **<<** "explicit spec. <int, double, 29>\n"**;**

**}**

**}**