**25.08.2023**

* copy elision
* C++17 (mandatory copy elision)
* Return Value Optimization RVO
* NRVO
* dinamik ömürü nesneler
* new ifadeleri
* delete ifadeleri
* operator new fonksiyonları
* operator delete fonksiyonları

PR Value bir nesne olmuyor C++17' ye göre

mesela

Myclass{}; //temp objects

**Temporary Materialization**

**class** Myclass

**{**

Myclass**(const** Myclass**&)** **=** **delete;**

**}**

**void** foo**(**Myclass x**)**

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

**{**

Myclass**{};** ***//PR Value***

Myclass x **=** Myclass**{};** ***//temporary materialization***

**const** Myclass **&**x **=** Myclass**{};** ***//temporary materialization***

foo**(**Myclass**{})**

foo**(**Myclass**{**46**})**

***/\****

***Myclass{} ve Myclass{46} bir nesne olmadığı için ctor ve dtor çağrılacak***

***copy ctor delete olmasına rağmen***

***\*/***

**}**

**Return Value Optimization (RVO)**

***// Return Value Optimization (RVO)***

**class** Myclass **{**

**public:**

Myclass

**{**

std**:** cout **<<** "default ctor\n"**;**

**}**

**~**Myclass**()**

**{**

std**:** cout **<<** "destructor\n"**;**

**}**

Myclass**(int)**

**{**

std**:** cout **<<** "Myclass(int)\n"**;**

**}**

Myclass**(const** Myclass**&)** **=** **delete;**

**};**

Myclass foo**(int** n**)**

**{**

***//code***

**return** Myclass**{**n**};**

**}**

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

**{**

Myclass m **=** foo**(**13**);** ***// int parametreli ctor çağrılır bir kez***

**}**

**Named Return Value Optimization (NRVO)**

std**::**string foo**()**

**{**

std**::**string str**(**1000**,** 'A'**);**

str **+=** "NNNNNN"**;**

**return** str**;**

**}**

main**()**

**{**

std**::**string s **=** foo**();**

**}**

Named Return Value Optimization (NRVO) ve Return Value Optimization (RVO), C++ programlamasında dönüş değeri optimize etme amacıyla kullanılan iki benzer tekniktir,

**RVO VS NRVO**

MyClass createObjectWithNRVO**()** **{**

MyClass obj**;**

**return** obj**;** ***// NRVO etkin olduğunda, bu nesne doğrudan dönüş değeri olarak kullanılır***

**}**

MyClass createObjectWithRVO**()** **{**

**return** MyClass**();** ***// RVO etkin olduğunda, bu nesne doğrudan dönüş değeri olarak kullanılır***

**}**

**class** Myclass **{**

**public:**

Myclass**()**

**{**

std**:** cout **<<** "default ctor\n"**;**

**}**

**-**Myclass**()**

**{**

std**:** cout **<<** "destructor\n"**;**

**)**

Myclass**(int)**

**{**

std**:** cout **<<** "Myclass(int)\n"**;**

**}**

Myclass**(const** Myclass**&)**

**{**

std**:** cout **<<** "copy ctor\n"**;**

**}**

Myclass**(**Myclass66**)**

**{**

std**:** cout **<<** "move ctor\n"**;**

**}**

Myclass**&** **operator=(const** Myclass**&)**

**{**

std**:** **:**cout **<<** "copy assignment\n"**;**

**return** **\*this;**

**}**

Myclass**&** **operator=((**Myclas**&&)**

**{**

std**:** cout **<<** "move assignment\n"**;**

**return** **\*this;**

**}**

**void** foo**()** **{};**

**void** bar**()** **{};**

**void** baz**()** **{};**

**};**

Myclass foo**()**

**{**

Myclass m**{**365**};**

std**::**cout **<<** "&m = " **<<** **&**m **<<** "\n"**;**

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

m**.**baz**();**

m**.**bar**();**

**return** m**;**

**}**

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

**{**

Myclass nec **=** func**();** ***// copy elision deniyor***

std**::**cout **<<** "&nec = " **<<** **&**nec **<<** "\n"**;**

***/\****

***move ya da copy ctor çağrılmadı sadece Myclass(int) ctor çağrıldı.***

***m{365} ile nec aynı nesnelerde aynı adreste tutuluyorlar***

***\*/***

**}**

**class** Myclass **{**

**public:**

std**::**string m\_str**;** ***// 10000;***

std**::**vector**<int>** m\_vec**;** ***// 2000;***

**};**

Myclass foo**();**

main**()**

**{**

Myclass m**;**

m **=** foo**();** ***// move assignment çağrılacak***

Myclass m1 **=** foo**();**

***// copy ve move ctor çağrılmaz***

***// mandatory copy elision olur***

***// copy elision olması için bir nesneyi hayata getirmemiz gerekiyor***

***/\****

***mandatory copy elision > move ctor > copy ctor***

***\*/***

**}**

copy elision olması için bir nesneyi hayata getirmemiz gerekiyor.

copy elision:

1) temporary object passing (mandatory)

2) returning a temporary object (mandatory)

3) returnnig an obejct of automatic storage class (optimization)

static storage class

* global nesneler
* static yerel nesneler
* sınıfların static veri elemanlar (static data members)

automatic storage class

* parameters
* local variabler

dynamic storage class

* new
* delete

thread-local storage class

**Dynamic Storage Class**

new Fighter --> bu ifadenin türü Fighter\*

void\* operator new(std::size\_t) standart’ta yapılan işlem

static\_cast<Fighter \*>(operator new(sizeof(Fighter)))->Fighter();

memory leak --> new operator ile alınan alanın geri verilmemesi

resource leak --> dtor ile yapılan işlemlerin yapılamaması (database bağlantısı kesilmemesi gibi)

**class** Myclass **{**

**public:**

Myclass **()**

**{**

std**::**cout **<<** "default ctor this = " **<<** **this** **<<** '\n'**;**

**}**

**~**Myclass**()**

**{**

std**::**cout **<<** "destructor this = " **<<** **this** **<<** '\n'**;**

**}**

**void** foo**)** **{};**

**void** bar**()** **{};**

**};**

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

**{**

Myclass **\***p **=** **new** Myclass**;**

Myclass **\***p1 **=** **new** Myclass**();**

Myclass **\***p2 **=** **new** Myclass**{};**

free**(**p**)** ***// undefined behavior***

**}**

delete p ifadesi aşağıdaki işlemleri yapar:

1. p->~Myclass();
2. operator delete(p);

**Array New**

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

**{**

std**::**cout **<<** "kac tam sayi:" **;**

std**::size\_t** n**;**

std**::**cin **>>** n**;**

**int** **\***p **=** **new** **int[**n**];**

**for** **(**std**::size\_t** i**{};** i **<** n**;** **++**i**)**

**{**

p**[**i**]** **=** i**;**

**}**

**for** **(**std**::size\_t** i**{};** i **<** n**;** **++**i**)**

**{**

std**:**cout **<<** p**[**i**]** **<<** " "**;**

**}**

**delete** **[]** p**;** ***// array delete***

**}**

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

**{**

std**::**cout **<<** "sizeof(Myclass) = " **<<** **sizeof(**Myclass**)** **<<** "\n"**;**

Myclass **\***p **=** **new** Myclass**[**10**];** ***// 10 tane Myclass nesnesi oluşur 10 ctor oluşur***

***// delete p undefined behavior***

**delete** **[]** p**;** ***// 10 dtor olur***

* sınıfların static veri elemanları
* sınıfların static üye fonksiyonları
* operator overloading
* namespace

**Sınıfların Static Veri Elemanları**

***// sınıfların static veri elemanları***

**class** Nec **{**

**static** **int** mx**;**

***/\****

***sınıfların static veri eleamnları staic anathar sözcüğü ile bildiriliyor***

***bu (tanımlama olmayan) (non-definind declaraion)***

***\*/***

**}**

***//global variable'lardan assembly açısında pek farkı yok***

main**()**

**{**

Nec**::**mx**;**

***// mx oluşturulan tüm nesneler için aynı mx aynı adreste yazılır***

Nec n1**,**n2**;**

n1**.**mx **=** 10 ***// n2.mx = 10 aynı şeydir***

**}**