**2023 08 04**

**Constexpr**

**constexpr** **int** x **=** 5**;** ***// x'in türü cont int***

**int** g**{};**

***// ilk ikisinin arasında hiçbir fark yok***

**constexpr** **int\*** p **=** **&**g**;**

**constexpr** **int** **const\*** p **=** **&**g**;**

***// yukarıdakilerden farklı***

**constexpr** **const** **int\*** p **=** **&**g**;**

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

**const** **int** x1 **=** foo**();** ***// gecerli***

**constexpr** **int** x2 **=** foo**();** ***// syntax hatası***

***//dizi olarak tanımlanabilir***

**constexpr** **int** primes**[]** **=** **{** 2**,** 3**,** 5**,** 7**,** 11**,** 13**,** 17**,** 19**,** 23**,** 29**};**

**constexpr** **int** x **=** primes**[**4**];**

**Constexpr Fonksiyonlar**

***1. Constexpr fonkisyon olması için belirli koşulları sağlaması gerekir:***

***- static ömürlü yerel değişkene sahip olmayacak***

***2. Eğer tüm parametrelerine sabit ifadeleri ile çağrı yapılırsa fonksiyonun***

***geri dönüş değeri derleme zamanında elde ediliyor.***

***eğer foo'ya verilen argumanların hepsi constexpr ise foo'nun geri dönüş değeri***

***derleme zamanında oluşur.***

***constexpr int x = foo(exp1, exp2, exp3)***

***Eğer argumanlar constexpr değilse run-time'da elde edilecek***

**constexpr** **int** ndigit**(int** x**)**

**{**

**if** **(**x **==** 0**)**

**return** 1**;**

**int** digit\_count**{};**

**while** **(**x**){**

**++**digit\_count**;**

x **/=** 10**;**

**}**

**return** digit\_count**;**

**}**

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

**{**

**int** a**[**ndigit**(**83123**)]{};** ***// compiler-time'da hesaplandı***

**const** **int** x **=** 3234**;**

**const** **int** y **=** 23234**;**

***// fonksiyonun parametresi sabit ifadesi***

**constexpr** **auto** val **=** ndigit**(**x **\*** y **-** 19**);** ***// compiler-time'da hesaplandı***

***// eğer syntax hatası verseydi artık sabit ifadesi bu diyemezdik***

**}**

**constexpr** **bool** isprime**(int** val**)**

**{**

**if** **(**val **<** 2**);**

**return** **false;**

**for** **(int** i**{** 7 **};** i **\*** i**;** i **<=** val**;** i**+=**2**)**

**{**

**if** **(**val **%** i **==** 0**)**

**return** **false;**

**}**

**return** **true;**

**}**

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

**{**

**const** **int** x **=** 12312**;**

**const** **int** y **=** 12312**;**

**constexpr** **auto** a **=** isprime**(**x **+** y **-** 19**);** ***// compiler-time***

**int** z **=** 123**;**

**constexpr** **auto** b **=** isprime**(**x **/** z **-** 12**);** ***// run-time***

**}**

NOT: constexpr fonkisyonlar headerda tanımlanır

**ODR (One Definition Rule)**

Değişkenler, sınıflar ve fonksiyonlar gibi yazılımsal bazı varlıkların bildirimleri program içinde birden fazla kez bulanabilir. Ancak tanımları tek olmak zorundadır.

**Inline Expansion**

Derleyici yazdığımız kodu fonksiyonun çağrıldığı yerde açar yani tüm kodu çağrıldığı noktaya eklenir. Bunu yapabilmesi derleyicinin fonksiyon kodunu görmesi gerekir.

Bunun avatanjıları:

* Linkleme maliyetini azaltır.
* Derleyicinin gördüğü kod statement artar ve optimazyonu artar

statement 1;

statement 2;

x = func(a, b); burda func kodu buraya eklenmiş gibi gözükür

statement 4;

statement 3;

derleyici inline expansion sayesinde func ve bütün statementları aynı anda görür ve optimazyonu ona göre yapar.

**Inline Anahtar Sözcüğü**

**inline** **int** foo**(int** x**,** **int** y**)** **{**

***/\****

***inline olarak tanımlansa bile derleyici inline expansion etmek zorunda değil.Aynı zamanda inline olarak tanımlanmasa bile derleyici bunu inline expansion***

***edebilir.***

***\*/***

**return** x **\*** 12 **\*** y**;**

**}**

***ahmet.cpp***

***inline int foo(int x, int y)***

***{***

***return x \* y - 5;***

***}***

***tunahan.cpp***

***inline int foo(int x, int y)***

***{***

***return x \* y - 5;***

***}***

***fonkisyonları inline olarak tanımlamasak ODR'a aykırı olurdu***

***\*/***

***// static ömürlü global ve class member***

**inline** **static** **int** g**{**12**};** ***// static ömürlüyü böyle tanımlayabiliriz.***

**Neleri başlık dosyasına koyarsam ODR ihlal etmemiş olurum.**

1. inline fonksiyon tanımları
2. inline değişken tanımları (C++17)
3. user-define type (class)
4. constexpr fonksiyonlar implictly inline

**C++ dilinde enumaration types**

***// modern cpp öncesi***

**enum** Color **{**Blue**,** Black**,** White**,** Purple**,** Red**};**

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

**{**

Color mycolor**;**

***// C'de geçerli***

mycolor **=** 3**;** ***// syntax hatası Cpp'da***

**}**

***/\****

***Modern olmayan C++ dilinde enum türlerinin istenmeyen***

***özellikleri :***

***1) underlying type derleyiciye bağlı olduğu için enum türleri başlık dosyalarında***

***incomplete type olarak kullanılamaz***

***enum Color;***

***struct Data {***

***Color mc; // syntax hatası olur çünkü Color sizeni bilmiyoruz***

***};***

***====================***

***2) Enum türleriden tamsayılar türüne örtülü dönüşüm olması***

***enum Color {Blue, Black, White, Purple, Red};***

***int main()***

***{***

***Color mycolor = Black;***

***int ival;***

***ival = mycolor;***

***}***

***====================***

***3)***

***// traffic\_ligth.h***

***enum TrafficLigth { Red, Yellow, Green};***

***// screen.h***

***enum ScreenColor{ Magenta, White, Black, Red};***

***isim çakışması bu header'lar syntax hatası yaratır.***

***\*/***