

C++

Ders 9

Ön İşlemciler

C++

Önişlemciler

Giriş

#include Önişlemci Direktifi

#define Önişlemci Direktifi: Sembolik Sabitler

#define Önişlemci Direktifi : Makrolar Şartlı

Derleme

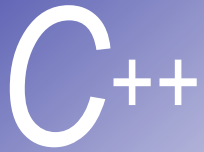
#error ve #pragma Önişlemci Direktifleri

ve ## Operatörleri

Satır Numaraları

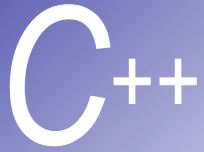
Önceden Tanımlı Sembolik Sabitler

Assert



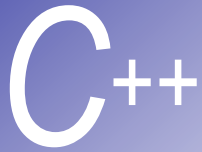
Giriş

- Önışlemler
 - Program derlenmeden önce meydana gelirler
 - Diğer dosyaların dahil edilmesi
 - *Sembolik sabitlerin tanımı ve makrolar,*
 - Program kodunun *şartlı derlenmesi*
 - *Önişlemci direktiflerin şartlı çalıştırılması*
- Önişlemci direktiflerinin formatı:
 - *# ile başlayan satırlar*
 - Satırlarda direktifler öncesi boşluk karakterler
 - C++ ifadesi değiller – noktalı virgöl (;) konmaz



#include Önışlemci Direktifi

- **#include**
 - Belirtilen dosyanın kopyası direktifin yerine yerleştirilmiştir
#include <filename> - dosya için standart kütüphaneyi arar (standart kütüphane dosyaları için kullanılır)
#include "filename" - öncelikle bulunulan dizinde arar, sonra standart kütüphanede arar (kullanıcı tanımlı dosyalar için kullanılır)
- Aşağıdaki durumlarda kullanılır
 - başlık dosyalarının yüklenmesinde (**#include <iostream>**)
 - birden fazla kaynak dosyayla birlikte derlenecek programlarda
 - Başlık dosyası - ortak deklarasyonlar ve tanımlar(sınıflar, yapılar, fonksiyon prototipleri)
 - Her dosyada **#include** ifadesi



#define Önışlemci Direktifi: Sembolik Sabitler

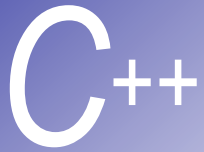
- **#define**
 - önışlemci direktifi sembolik sabitler ve makrolar oluşturmak için kullanılır.
- **Sembolik Sabitler**
 - Program derlendiği zaman, tüm sembolik sabitlerin olduğu yerler atama teksti ile yer değiştirilir
- **Format:**
`#define belirteç atanacak-text`
 - Örnek:

```
#define PI 3.14159
```

 - Belirtecın sağındaki herşey yerleştirilecek text olarak kullanılır

```
#define PI = 3.14159
```

 - "PI" " = 3.14159" ile yer değiştirir, muhtemelen hata ile sonuçlanır
 - sembolik sabitler daha fazla `#define` ifadeleri ile tekrar tanımlanamamazlar.



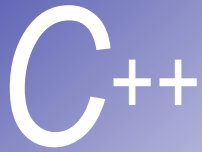
#define Önışlemci Direktifi: Macrolar

- Macro – `#define`’ da tanımlanan operasyon
 - C programları için tasarlanmıştır
 - argümanlırsız makro: sembolik sabitlerdeki gibi davranılır
 - argümanlı makro: argümanlar yer değıştirilecek metnin yerini alır, macro genişletilir
 - Text yer değıştirme işlemi yapar– veri türü kontrolü olmaz

Örnek:

```
#define CIRCLE_AREA( x ) ( PI * ( x ) * ( x ) )
```

```
area = CIRCLE_AREA( 4 ); aşağıdaki duruma gelir  
area = ( 3.14159 * ( 4 ) * ( 4 ) );
```



#define Önışlemci Direktifi: Makrolar (II)

- Parantez kullanımı:

- parantezsiz,

```
#define CIRCLE_AREA( x )  PI * ( x ) * ( x )  
area = CIRCLE_AREA( c + 2 );
```

aşğıdaki gibi olur

```
area = 3.14159 * c + 2 * c + 2;
```

ve bu hesap yanlıştır

- çoklu argümanlar:

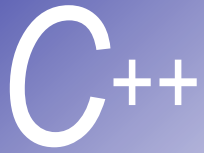
```
#define RECTANGLE_AREA( x, y )  ( ( x ) * ( y ) )  
rectArea = RECTANGLE_AREA( a + 4, b + 7 );
```

aşğıdaki gibi olur

```
rectArea = ( ( a + 4 ) * ( b + 7 ) );
```

- #undef

- Daha sonra tanımlanabilecek sembolik sabit ve macroyu tanımsız hale getirir

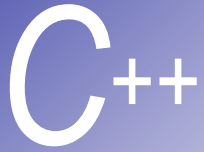


Şartlı Derleme

- Şartlı Derleme
 - Önilemci direktiflerini ve derlemeyi kontrol eder
 - Tür değıştirme ifadeleri , **sizeof** , numaralandırılmış sabitler hesaplanamaz
- **If' e benzer yapı**

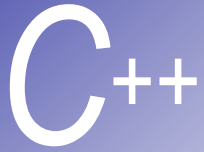
```
#if !defined( NULL )
    #define NULL 0
#endif
```

 - Sembolik değışken **NULL**'un tanımlanmışlığını belirler
 - Eğer **NULL** tanımlı ise , **defined(NULL)** , 1 değerini alır
 - eğer **NULL** tanımlı değilse, **NULL**' u 0 olarak tanımlar
 - tüm **#if**' ler **#endif** ile biter
 - **#ifdef** , **#if defined(name)**'in kısaltılmış şeklidir
 - **#ifndef** **#if !defined(name)**'in kısaltılmış şeklidir



Şartlı Derleme (II)

- Diğer İfadeler:
 - `#elif` - `if` yapısında `else if` 'in aynısıdır
 - `#else` - `if` yapısında `else`'in aynısıdır
 - "Comment out" kodu
 - `/* ... */` kullanılamaz
 - Aşağıdaki kullanılır
 - `#if 0`
 - Yorum içine alınacak kod
 - `#endif`
- kod kullanılmak isteniyorsa `0`, `1` 'le değiştirilmeli



Şartlı Derleme (III)

- Hatadan arındırma

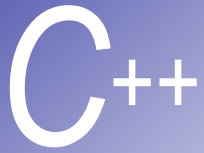
```
#define DEBUG 1
```

```
#ifdef DEBUG
```

```
    cerr << "Variable x = " << x << endl;
```

```
#endif
```

DEBUG'un tanımlanması kodu çalışır kılar. Kod doğrulandıktan sonra, #define ifade silinir ve hata arındırma ifadeleri göz ardı edilir



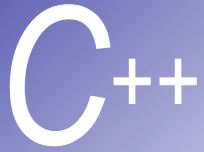
#error ve #pragma Önişlemci Direktifi

■ **#error** *tokenları*

- token – boşluklarla ayrılmış ard arda gelen karakter dizileri
 - "I like C++" 'da üç token vardır
- mesajları ve tokenları yazar (uygulamaya bağlıdır)
- örneğin: ne zaman **#error** ile karşılaşılsa, token gösterilir ve önişlemler durur (program derlenmez)

■ **#pragma** *tokenları*

- Uygulama tanımlı faaliyet (derleme dokümantasyonuna yardımcı olur)
- Derleyici tarafından tanınmayan pragma' lar ignore edilir



ve ## Operatörleri

- # - atama text tokenı çift tırnaklı diziye dönüştürülür

```
#define HELLO( x ) cout << "Hello " << x << endl;
```

#'ye dikkat

HELLO(John) becomes

```
cout << "Hello, " << "John" << endl;
```

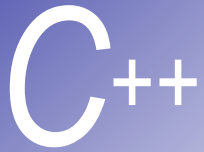
- Boşluklarla ayrılmış diziler cout kullanılarak birleştirilir

- ## - iki tokenı birleştirir

```
#define TOKENCONCAT( x, y ) x ## y
```

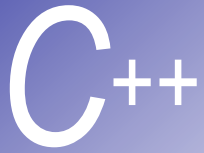
TOKENCONCAT(O, K)

OK şeklini alır



Satır Numaraları

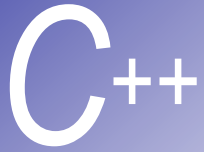
- **#line**
 - Ardışık kod satırlarını tamsayı değerden başlayarak numaralandırır
 - dosya ismi içerilebilir
- **#line 100 "myFile.c"**
 - sonraki kaynak kod dosyasından başlanarak satırlar **100** üzerinden numaralandırılır
 - Hata amaçlı, dosya ismi **"myFile.c"** dir
 - hataları daha anlamlı kılar
 - satır numaraları kod dosyada belirmez



Önceden Tanımlı Sembolik Sabitler

- Beş önceden tanımlı sembolik sabit vardır
 - **#define** veya **#undef** \ da kullanılamazlar

Sembolik Sabit	Tanımı
__LINE__	bulunulan kaynak kod satır satır numarası (bir tamsayı sabit).
__FILE__	Kaynak dosyanın varsayılan ismi (bir dizi).
__DATE__	Kaynak dosyanın derlendiği tarih (" Mmm dd yyyy " , " Jan 19 2001 " gibi).
__TIME__	Kaynak dosyanın derlendiği zaman (" hh:mm:ss " formundaki dizi).



Assert

- **Assert** makrosu
 - `<cassert>` başlık dosyası
 - İfadelerin değerlerini test eder
 - eğer 0 (false) ise hata mesajı yazılır ve **abort** çağrılır
- `assert(x <= 10);`
- eğer **NDEBUG** tanımlıysa, bütün sonraki **assert** ifadeler göz ardı edilir
 - `#define NDEBUG`