

C Preprocessor Directives

- `#include`
- `#define`
- `#undef`
- `#ifdef`
- `#ifndef`
- `#if`
- `#else`
- `#elif`
- `#endif`
- `#error`
- `#pragma`

- | | |
|------------------------------|---------------------------------------------|
| ▪ Object-like Macros | ▪ Function-like Macros |
| <code>#define PI 3.14</code> | <code>#define circleArea(r) (PI*r*r)</code> |

Obje tipi makrolar ve **fonksiyon tipi makrolar** bulunmaktadır. Obje tipinde `PI = 3.14`, fonksiyon tipinde ise daire formülü direkt kullanılabilir.

#include

```
#include "dosya_adı" ya da #include <dosya_adı>
```

`#include` emri, daha önceden hazırlanan, standart veya kullanıcı tarafından tanımlanan dosyalarda saklanan, sık kullanılan veri ve fonksiyon bildirimlerini programa dahil etmede kullanılır. Böyle dosyalara *başlık dosyaları* adı verilir ve isimleri `dosyaadi.h` şeklinde olur.

```
#include <stdio.h>
#include <stdlib.h>
```

Önişlemciye `#include` satırının yerini, belirtilen dosyanın alacağını gösterir. Eğer `dosya_adı` çift tırnak içine alınmışsa, o zaman önişlemci, dosyayı kaynak program dosyasının saklandığı altdizinde arar. Eğer `dosya_adı` açılı parantezler içinde ise, o zaman dosya, böyle `#include` emirleri için aramaların yapıldığı standart altdizin(ler)de aranır.

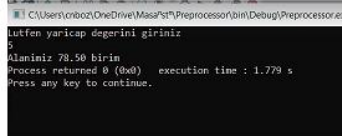
#define

```
#define tanıtıcı_sözcük karakter_dizisi
```

Bu tür bir `#define` emri, emirden sonra gelen program deyimlerinde `tanıtıcı_sözcük` bulunan her yerde, onun yerine `karakter_dizisinin` konulacağını gösterir. Örneğin,

#define XYZ 100

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define PI 3.14
4 #define circleArea(r) (PI*r*r)
5
6
7 int main()
8 {
9     float yaricap, alan;
10    printf("Lutfen yaricap degerini giriniz\n");
11    scanf("%f", &yaricap);
12    alan=circleArea(yaricap);
13    printf("Alanimiz %.2f birim", alan);
14
15    return 0;
16 }
17
```



```
C:\Users\cnboz\OneDrive\Masaüstü\Preprocessor\bin\Debug\Preprocessor.exe
Lutfen yaricap degerini giriniz
78.56
Alanimiz 1950.00 birim
Process returned 0 (0x0)   execution time : 1.779 s
Press any key to continue.
```

#undef

#undef *tanıtıcı_sözcük*

Daha önce tanımlanmış bir makronun tanımını kaldırmak için **#undef** emrini, arkasına makro ismini belirterek, kullanabilirsiniz.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define PI 3.14
4 #undef PI
5
6
7
8 int main()
9 {
10
11    printf("%f", PI);
12
13    return 0;
14 }
15
```

error: 'PI' undeclared (first use in this function)

#ifdef

#if, **#ifdef**, **#ifndef**, **#else**, **#elif** ve **#endif** emirleri, bazı koşullara bağlı olarak derlemeyi kontrol etmek için kullanılırlar.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #define PI 3.14
4
5  /*#ifdef PII
6  //successful code
7  #else
8  //else code
9  #endif */
10
11
12 int main()
13 {
14
15     #ifdef PII
16         printf("C harika gidiyor");
17     #else
18         printf("C muthis gidiyor");
19     #endif
20
21
22     return 0;
23 }

```

C:\Users\cnboz\OneDrive\Masaüstü\Preprocessor\bin\Debug\Preprocessor.exe
C muthis gidiyor
Process returned 0 (0x0) execution time : 0.061 s
Press any key to continue.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #define PI 3.14
4
5  /*#ifdef PII
6  //successful code
7  #else
8  //else code
9  #endif */
10
11
12 int main()
13 {
14
15     #ifdef PI
16         printf("C harika gidiyor");
17     #endif
18
19
20
21
22     return 0;
23 }

```

C:\Users\cnboz\OneDrive\Masaüstü\Preprocessor\bin\Debug\Preprocessor.exe
C harika gidiyor
Process returned 0 (0x0) execution time : 0.058 s
Press any key to continue.

#if/else


#if, #ifdef, #ifndef, #else, #elif ve #endif emirleri, bazı koşullara bağlı olarak derlemeyi kontrol etmek için kullanılırlar.

#if *değişmez ifade*

emri, eğer *değişmez_ifadenin* değeri doğru (sıfırdan farklı) ise bir sonraki #endif veya #else (veya #elif) emrine kadar olan satırların derlemeye dahil edilmesini sağlar. İlk durumda, derleme #endif'ten sonra, normal olarak devam eder; ikinci durumda ise, #else'den sonra ilgili #endif'e kadar olan satırlar önışlemci tarafından atlanır (yani derleyiciye gönderilmez). Eğer *değişmez_ifadeni* değeri yanlış (0L) ise, #if ile #endif veya #else (hangisi önce geliyorsa) arasındaki metin atlanır. #if emirlerinin içiçe yazılabildiğine dikkat edin, bunun için hangi #else'in hangi #if'e ait olduğunu bilmemiz gerekir: #else bir #endif ile bağlantısı olmayan, yukarıdaki en yakın #if'e aittir.

Değişmez_ifade, önışlemcinin önışlem zamanında hesaplayabileceği herhangi bir C ifadesidir; **sizeof** ve kalıp işleçleriyle sayım ve kayan noktalı sayı değişmezleri burada kullanılamazlar. Daha önce önışlemciye tanımlanmış makrolar ise kullanılabilirler; ancak, tanımsız isimlerin yerine **0L** konulur. *Değişmez_ifade* her zaman **long** duyarlılıkta hesaplanır.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define PI 3.14
4  #define SAYI 5
5
6
7  int main()
8  {
9
10
11      #if SAYI == 5
12          printf("Sayimiz : %d", SAYI);
13      #endif
14
15
16
17      return 0;
18  }
```



```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define PI 3.14
4  #define SAYI 5
5
6
7  int main()
8  {
9
10
11      #if(SAYI == 6)
12          printf("Sayimiz : %d", SAYI);
13      #else
14          printf("Sayimiz 5 den farklıdır");
15      #endif
16
17
18
19      return 0;
20  }
```



#error

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define PI 3.14
4 #define SAYI 5
5
6
7
8 #if !defined POWER || POWER < 2
9 #error POWER is not defined or is less than 2
10 #endif
```

C:\Users\cn... 8 error: #error POWER is not defined or is less than 2

PREDEFINED MACROS

__DATE__: Programın kaynak kodunun nesne koduna dönüştürüldüğü tarihi verir. Basitçe söylemek gerekirse, programın derlendiği tarihi döndürür. Tarih, mm yy yy yy biçimindedir .

__FILE__: Bilgisayarda o anda çalışmakta olan programın dosya adını tutar. Ayrıca hata ayıklama, hata raporları ve günlük mesajları oluşturmada da kullanılır.

__LINE__: Derlemedeki programın geçerli satır numarasını içerir. Çağrıldığı hat numarasını verir. Günlük ifadeleri, hata mesajları, istisnalar ve hata ayıklama kodları oluşturmak için kullanılır.

__STDC__: Derleyici standardını doğrulamak için kullanılır. Genel olarak derleyicinin ISO Standardı C'ye uygun olduğu anlamına gelen 1 değerini tutar.

__TIME__: programın derlendiği zamanı verir. Saat, saat: dakika :saniye biçimindedir .

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define PI 3.14
4 #define circleArea(r) (PI*r*r)
5
6 int main()
7 {
8     printf("File :%s\n", __FILE__ );
9     printf("Date :%s\n", __DATE__ );
10    printf("Time :%s\n", __TIME__ );
11    printf("Line :%d\n", __LINE__ );
12    printf("STDC :%d\n", __STDC__ );
13    return 0;
14 }
15
```

C:\Users\cnboz\OneDrive\Masaüstü\Preprocessor\bin\Debug\Preprocessor.exe

File :C:\Users\cnboz\OneDrive\Masaüstü\Preprocessor\main.c
Date :May 26 2022
Time :09:37:58
Line :11
STDC :1

Process returned 0 (0x0) execution time : 0.040 s
Press any key to continue.