Giriş

Bir C (veya C++) programlama dilinde, program başında diyez ('#') işareti ile başlayan satırlar geçekte C (veya C++) diline ait olmayıp ön işlemci dilidir. Bu yüzden derleme işlemleri iki adımda yapılır. Daha ayrıntılı bilgi için bkz: Bölüm 22.

Makro bildirimleri veya Yönergeleri (direktive) derleme öncesi komutlarıdır. Bunlar tipik olarak:

- programları değiştirmek
- program parçalarını kaynak programında birleştirmek
- derleme sırasında bazı uyarı mesajlarını aktif veya pasif hale getirmek

için kullanılır. Genelde makro bildirimleri kaynak dosyaların en başında verilir.

C dilinde kullanılan Yönergeler (önişlemci komutları) şunlardır:

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

20.1 `#include` Yönergesi

Bu önişlemci verilen dosyanın içeriğini, kullanıldığı yerde kaynak dosyasının içine ekler. Çoğunlukla derleyiciye ait komut kütüphanelerinde bulunan fonksiyonların prototiplerinin ve diğer çeşitli tanımlamaların bulunlunduğu (h uzantılı) başlık dosyalarının programa dahil edilmesinde kullanılır[2]. İki tür kullanımı vardır:

```
#include <dosya_adı.h>
```

veya

```
#include "dosya_adı.h"
```

- Birinci kullanımda dosyanın nerede bulunduğu derleyici için verilen ulaşım yolu ile belirlenir. Bu yol genellikle include dizini ile son bulur. Başlık dosyalarının saklandığı include dizini
  - Borland firmasına ait Turbo C derleyicisinde : `C:\TC\INCLUDE`
  - Linux ortamında : `/usr/include` şeklindedir.
- İkinci kullanımlada dosyanın bulunduğu yer aktif dizin olarak kabul edilir. Aksi halde yol tam olarak verilmelidir.

`#include` deyimi ile program ilave edilecek dosya C fonksiyonları içerebileceği gibi basit deyimler de içerebilir. Bunun için bir sınırlandırma yoktur. Hatta uzantıları `.h` olması bile gerekmez. Program 20.1 ve Program 20.2'yi inceleyin.

Program 20.1: *#include önişlemcisinin kullanımı için bir örnek*

```
01: /* 20prg01.c: faktoriyel ve kombinasyon
02: hesaplamaları */
03:
04: #include <stdio.h>
05:
06: #include "komb.h"
07:
08: int main()
09: {
10:     int i;
11:
12:     /* 0 dan 10 yekadar olan sayıların faktoriyelleri
13: */
14:     for(i = 0; i<=10; i++)
15:        printf("%2d! = %d\n",i,faktoriyel(i));
16:
17:     /* 10'un 1li, 2li, ... kombinasyonları */
18:     for(i = 0; i<=10; i++)
19:        printf("C(10,%2d) = %d\n",i,C(10,i));
20:
     return 0;
    }
```

ÇIKTI

```
 0! = 1
 1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5040
 8! = 40320
 9! = 362880
10! = 3628800
C(10, 0) = 1
C(10, 1) = 10
C(10, 2) = 45
C(10, 3) = 120
C(10, 4) = 210
C(10, 5) = 252
C(10, 6) = 210
C(10, 7) = 120
C(10, 8) = 45
C(10, 9) = 10
C(10,10) = 1
```

Program 20.1'e 5.satırda `#include` önişlemcisi ile komb.h adlı başlık dosyası eklenmiştir. komb.h faktoriyel ve kombinasyon işlemleri için fonksiyonlar barındırır. Bu dosyanın içeriği şöyledir:

```
/* komb.h
   Kombinasyon işlemleri ile ilgi fonksiyonlar
*/

/* n! sayısını gönderir */
int faktoriyel(int n){
```

```
  int i, f;

   for(f=1, i=2; i<=n; i++)
      f *= i;

   return f;
}

/* n'nin r'li kombinasyonunu hesaplar */
int C(int n, int r){
  return (
faktoriyel(n)/(faktoriyel(r)*faktoriyel(n-r)) );
}
```

Program 20.2: *#include önişlemcisinin kullanımı için başka bir örnek*

```
01: /* 20prg02.c:
02:    Klavyeden girilen bir doğal sayının, kaç basamaklı
03:    olduğunu bulup ekrana yazar.
04: */
05:
06: #include <stdio.h>
07:
08:
09: int main(){
10:
11:   #include "bildirim.inc"
12:
13:   printf("Bir dogal sayi gir: ");
14:   scanf("%d",&sayi);
15:
16:
17:   if(sayi<0) return BASARISIZ;
18:
19:   b = 0;
20:   n = sayi;
21:
22:   while(n>0){
23:     n /= 10;
24:     b++;
25:   }
26:
27:   printf("%d  %d basamakli bir sayidir.\n",sayi,b);
28:
29:   return BASARILI;
30: }
```

Dikkat edilirse Program 20.2 içinde değişkenler tanımlanmamıştır. 11. satırdaki bildirim.inc dosyası değişken bildirimlerini barındırmaktadır. Bu özellik (veya esneklik), çok büyük ve profesyonel programlarda kullanılmaktadır.

ÇIKTI

```
Bir dogal sayi gir: 4578
4578  4 basamakli bir sayidir.
```

20.2 `#define` Yönergesi

Bu önişlemci komutu, kaynak dosyada bir isim yerine başka bir isimin yerleştirilmesini sağlar. Programda kullanılan bu *sembolik* isimler başta ana program olmak üzere bütün alt programlarda da aynı değere sahiptir. Yani `#define` önişlemcisi ile tanımlanan her ne olursa olsun, tanımlama bütün fonksiyonlarda kullanılabilir. Bir çeşit genel (global) bildirim gibi davranır. Örneğin:

Program 20.3: *#define önişlemcisinin kullanımı*

```
01:  /* 20prg03.c: #define önişlemcisinin kullanımı */
02:
03:  #include <stdio.h>
04:
05:  #define PROGRAM main()
06:  #define BASLA   {
07:  #define BIT     }
08:  #define YAZ     printf
09:
10:  PROGRAM
11:  BASLA
12:     YAZ("Merhaba C!..\n");
13:  BIT
14:
```

Program 20.3 derleme işleminden önce `#define` ile verilen ilk sembolik isimler yerine ikinci isimler yerleştirildikten sonra program aşağıdaki durmuma gelir:

```
/* 20prg03.c: #define önişlemcisinin kullanımı */

#include <stdio.h>

main()
{
  printf("Merhaba C!..\n");
}
```

Bu önişlemciyi kullanak sembolik sabitler tanımlamak mümkündür. Örneğin:

```
#define PI      3.1415926
#define IKI_PI  2.0*PI
#define YUZ     100
```

gibi.

`#define` önişlemcisinin kullanımı için iyi bir örnek Program 20.4 de verilmiştir. Program km/s biriminde verilen bir hızı m/s birimine çevirir[4].

Program 20.4: *#define önişlemcisinin kullanımı*

```
01:   /* 20prg04.c: km/s biriminde verilen hızı m/s
02:  cinsinden hesaplar */
03:
04:   #include <stdio.h>
05:
06:   #define km   *1000.0
07:   #define saat *3600.0
08:
09:   main()
```

```
10:   {
11:      double yol,zaman,hiz;
12:
13:      yol   = 100 km;
14:      zaman = 1.2 saat;
15:
16:      hiz = yol/zaman;
17:
18:      printf("HIZ = %lf m/s\n",hiz);
19:
      }
```

ÇIKTI

```
HIZ = 23.148148 m/s
```

5. ve 6. satırda tanımlanan sembolik sabitler `km` ve `saat` program içinde kullanıldığında sol taraflarındaki sayıyı sırasıyla 1000 ve 3600 ile çarparlar. 12. satırdaki `yol` değişkenine 100*1000.0 değeri atanır. Benzer olarak 13. satırdaki `zaman` değişkenine 1.2*3600.0 sayısı atanır. Dikkat edilirse sembolik sabitler kullanıldığında programın okunurluğu artmakta ve bundan dolayı hata ayıklama kolaylaşmaktadır.

`#define` önişlemcisi ile parametrik tanımlamalar veya global fonksiyonlar tanımlamak mümkün olur. Örneğin:

Program 20.5: *Makro fonksiyon tanimlama*

```
01: /* 20prg05.c: Makro fonksiyon tanimlama. */
02:
03: #include <stdio.h>
04: #include <math.h>
05:
06: /* makro fonksiyonlar */
07: #define kare(x)   (x*x)
08: #define topl(x,y) (x+y)
09: #define carp(x,y) (x*y)
10: #define hipo(x,y) sqrt(x*x+y*y)
11:
12: main(void)
13: {
14:    float a=3.0, b=4.0;
15:
16:    printf("kare(2)   = %f\n",kare(2));
17:    printf("topl(a,b) = %f\n",topl(a,b));
18:    printf("carp(a,b) = %f\n",carp(a,b));
19:    printf("hipo(a,b) = %f\n",hipo(a,b));
20: }
```

ÇIKTI

```
kare(2)   = 4.000000
topl(a,b) = 7.000000
carp(a,b) = 12.000000
hipo(a,b) = 5.000000
```

Programda tanımlanan `kare(2)` ifadesi `(2)*(2)` şeklinde yorumlar. Benzer durum diğer makrolar için de geçerlidir.

Makrolar C'de çok sık kullanılır. Örneğin, tek boyutlu bir dizinin boyutu öğrenilmek istendiğinde aşağıdaki makro kullanılabilir:

```
#define BOYUT sizeof(DIZI)/sizof(DIZI[0])
...
int a[10], n;
...
n = BOYUT(a);
```

Son satırdaki işlemle, `n` değişkeninine (`a` dizisinin boyutu) 10 değeri atanır.

İşte ilginç bir makro daha. Daha önce anlatılan `takas(a,b)` fonksiyonu gösterici kullanmadan aşağıdaki makro ile yazılabilir:

```
#define takas(x,y) {g=(x); (x)=(y); (y)=g;}
...
int x=22, y=33, g;        /* g geçici bir değişken */
...
printf("%d %d\n",x,y);    /* 22 33 */

takas(a,b)

printf("%d %d\n",a,b);    /* 33 22 */
...
```

20.3 `#undef` Yönergesi

`#define` ile tanımlanan bir isim, orjinal tanımlamaları kaldırmaksızın farklı değerler için tekrar tanımlanamaz.

```
#define SIFRE 14576  /* ilk tanimlama */
 ...
#define SIFRE 22357  /* hata! tanımlama tekrarlandı. */
```

Eğer `#define` ile tanımlanan bir ifade yeniden tanımlanmak istenirse, `#undef` önişlemcisi ile önceki tanımlama iptal edildikten sonra `#define` ile yenisi değiştirilir. Yani:

```
#define SIFRE 14576  /* ilk tanimlama */
 ...
#undef  SIFRE        /* ilk tanımlamayı iptal et */
#define SIFRE 22357  /* yeni tanımlama */
```

20.4 `#if, #elif, #else ve #endif` Yönergeleri

Bu önişlemciler, makro düzeyinde kontrol deyimleridir. Genel kullanım biçimi:

```
#if (ifade1)
       tanımlama blogu1
#elif (ifade2)
       tanımlama blogu2
...
#else
```

```
        tanımlama bloguN
#endif
```

şeklindedir. Burada:

- `#if` makrosu `if` deyimine
- `#elif` makrosu `else if` deyimine
- `#else` makrosu `else` deyimine
- `#endif` makrosu `if` deyiminin sonuna

karşılık gelmektedir. Bu makrolar, donanıma veya işletim sistemine uygun olarak değişik makroların tanımlanmasına izin verir. Örneğin:

Program 20.6: *Kontrol önişlemcilerinin kullanımı*

```
01: /* 20prg06.c: Kontrol ön işlemcilerinin kullanımı */
02:
03: #include <stdio.h>
04:
05: #if(sizeof(int)==2)
06:   #define ISLETIM_SISTEMI  "16 bitlik isletim
07: sistemi."
08: #else
09:   #define ISLETIM_SISTEMI  "32 bitlik isletim
10: sistemi."
11: #endif
12:
13: int main()
14: {
15:    printf(ISLETIM_SISTEMI);
16:
17: return 0;
    }
```

ÇIKTI

```
32 bitlik isletim sistemi.
```

Bu program eski DOS işletim siteminde derlenip çalıştırıldığında, program çıktısı şöyle olur:

ÇIKTI

```
16 bitlik isletim sistemi.
```

20.5 `#ifdef` ve `#ifndef` Yönergeleri

- `#ifdef` önişlemcisi ile, bir ismin *tanımlanmış* olup olmadığı
- `#ifndef` önişlemcisi ile, bir ismin *tanımlanmamış* olup olmadığı

sorugulanır. Örneğin:

```
#ifndef SIFRE
   #define SIFRE 22357
#endif
```

gibi.

Program 20.7: *Tanımlanmış ise pi sayısını kullanır.*

```
01: /* 20prg07.c: Tanımlanmış ise PI sayısını kullanır */
02:
03: #include <stdio.h>
04: #include <math.h>
05:
06: #define PI 3.141593
07:
08: main()
09: {
10:    double c, r = 21.3;
11:
12:     #ifdef PI
13:        c = 2.0 * PI * r;
14:        printf("Dairenin cevresi = %lf\n",c);
15:     #else
16:        printf("PI saysisi tanimlanmamis.\n");
17:     #endif
18: }
```

ÇIKTI

```
Dairenin cevresi = 133.831862
```

20.6 #error Yönergesi

Önişlemci bu deyimle karşılaşınca yanındaki mesajı ekrana yazar ve derleme işlemine son verir. Mesela, yazmış olduğunuz program 32 bitlik bir işletim sistemi (WINDOWS veya Linux gibi) için tasarlanmışsa ve program 16 bitlik işletim sisteminde (MSDOS gibi) derlenecekse kullanıcıya buna dair bir uyarı mesajı vermek uygun olur[2-4]. Örneğin:

```
#if (sizeof(int)==2)
   #error Bu program 16 bitlik işletim sisteminde derlenemez !...
#endif
```

Eğer DOS altında çalışıyorsanız önişlemci derleme işlemine:

```
Bu program 16 bitlik işletim sisteminde derlenemez !...
```

mesajı ile son verir. Mesajın tırnak içine alınmadığına dikkat ediniz.

20.7 Önceden Tanımlanmış Sembolik Sabitler

Bazı sembolik sabitler derleyici tarafından önceden tanımlanmıştır. Bu sabitlerden bazıları Tablo 20.1 de verilmiştir.

Tablo 20.1: *Önceden tanımlı bazı sembolik sabitler*

Sabit ismi	Açıklama
`__LINE__`	Önişlemci bu sabit yerine kaynak koddaki o anda bulunan satır numarasını yerleştirir.
`__FILE__`	Kaynak dosyanın ismin tutar.
`__DATE__`	Önişlemci bu sabit yerine derlemenin yapıldığı zaman tarihi (ay gün yıl formatında) yazar.
`__TIME__`	Önişlemci bu sabit yerine derlemenin yapıldığı zaman zamanı (sa:dak:sn gün yıl formatında) yazar.
`__STDC__`	C dilinde kullanılan kimi anahtar sözcükler standart değildir. Derleyici eğer yalnızca standart C'nin anahtar sözcüklerini destekliyorsa bu sabit tanımlı varsayılır.
`M_PI`	Pi sayısını tutar (`M_PI = 3.14159265358979323846`). Ayrıca bkz: `math.h`
`M_E`	e sayısını tutar (`M_E = 2.7182818284590452354`). Ayrıca bkz: `math.h`
`RAND_MAX`	Rastgele sayı üretec fonksiyonu `rand()` ile döndürlen en büyük sayıyı tutar. (32 bit işletim sitemi için: `RAND_MAX = 2147483647`). Ayrıca bkz: `stdlib.h`

Aşağıdaki örnekleri inceleyiniz:

Program 20.8: *C dilindeki bazı tanımlı sabitler*

```
01: /* 20prg08.c: Sembolik sabitler */
02:
03: #include <stdio.h>
04:
05: main()
06: {
07:    printf("Satir No   : %d\n",__LINE__);
08:    printf("Dosya adi  : %s\n",__FILE__);
09:    printf("Tarih      : %s\n",__DATE__);
10:    printf("Saat       : %s\n",__TIME__);
11: }
```

ÇIKTI

```
Satir No   : 7
Dosya adi  : 20prg08.c
Tarih      : Sep 21 2008
Saat       : 01:58:56
```

Program 20.9: *C dilindeki bazı tanımlı sabitler*

```
01: /* 20prg09.c: Sembolik sabitler */
02:
03: #include <stdio.h>
04: #include <math.h>
05: #include <stdlib.h>
06:
07: #ifndef __STDC__
08:   #error Bu derleyici ANSI C degil.
09: #endif
10:
11: #ifndef RAND_MAX
12:   #error RAND_MAX tanımlı degil.
13: #endif
14:
15: main()
16: {
17:    double r = (double) rand()/RAND_MAX;
18:    double ikiPi = 2.0*M_PI;
19:    double birBoluE = 1.0/M_E;
20:
21:    printf("r  = %lf\n",r);
22:    printf("ikiPi = %lf\n",ikiPi);
23:    printf("birBoluE = %lf\n",birBoluE);
24: }
```

ÇIKTI

```
r  = 0.840188
ikiPi = 6.283185
birBoluE = 0.367879
```