PROGRAMLAMA

DERS NOTLARI

DERS 1: *GİRİŞ*

Giriş

Bu ilk derste, bir C programın nasıl derlenip çalıştırılacağı ve Internet'te bulabileceğiz derleyicilerden bahsedilecektir. En basit C programının derleyip çalıştırdıktan sonra, geriye kalan sadece C Programlama Dili'nin kurallarını, yapısını ve deyimlerini öğrenmekten ibarettir.

1.1 Tarihçe

C Programlama Dili genel amaçlı orta seviyeli ve yapısal bir programlama dilidir. 1972 yılında Dennis Ritchie tarafından Bell Telefon Labaraturvarında Unix işletim sistemi ile kullanılmak için tasarlanmıştır. C, özellikle sistem programlamada sembolik makine dili (Asembler) ile tercih edilmektedir. İşletim sistemleri, derleyiciler ve debug gibi aşağı seviyeli sistem programlarının yazılımında yoğun olarak C programlama dili kullanılır.

C'nin yayılması ve gelişmesi, büyük bir bölümü C dili ile yazılan UNIX işletim sisteminin popüler olmasıyla başlamıştır. C Programlama Dili, hemen her alanda kullanılmaktadır. Günümüzde nesneye yönelik programlama dilleri (C++, Java) ve script dilleri (JavaScript, JavaApplet, PHP) gibi programlama dilleri C Programlama Dili'nden esinlenmiştir.

C taşınabilir (portable) bir dildir. Yani herhangi bir C programı hiçbir değişikliğe uğramadan, veya çok az bir değişimle, başka bir derleyicide ve/veya işletim sisteminde derlenebilir. Örneğin, Windows işletim sistemlerinde yazılan bir C kodu, Linux, UNIX veya VAX gibi işletim sistemlerinde de derlenebilir. Taşınabilirlik, herkesin kabul ettiği bir standart ile gerçekleştirilebilir. Bugün, C Programla Dili için **American National Standards Institute (ANSI)** kurumunun Mart 2000'de belirlediği **C99: ISO/IEC 9899:1999** standartı **Standart C**olarak kabul edilmiştir.

Burada verilen C notarında, ağırlıklı olarak ANSI C veya diğer adıyla Standart C konu edilmiştir.

1.2 Neden C?

- C, en popüler dildir. Bkz: langpop.com
- C, güçlü ve esnek bir dildir. C ile işletim sistemi veya derleyici yazabilir, kelime işlemciler oluşturabilir veya grafik çizebilirsiniz.

- C, iyi bir yazılım geliştirme ortamına sahiptir.
- C, özel komut ve veri tipi tanımlamasına izin verir.
- C, taşınabilir bir dildir.
- C, gelişimini tamamlamış ve standardı oluşmuş bir dildir.
- C, yapısal bir dildir. C kodları *fonksiyon* olarak adlandıralan alt programlardan oluşmuştur.
- C++, Java, JavaScript, JavaApplet, PHP, C#, ... gibi diller C dilinden esinlenmiştir.

1.3 İlk C Programı

Program 1.1 de verilen C programı derlendikten sonra, ekrana '`Merhaba Dünya!`' yazısını basan yalın bir C programıdır. Satır başlarına yerleştirilen 1:, 2: 3: ... rakamlarının yazılmasına gerek yoktur. Bu rakamlar sadece daha sonra program ile ilgili açıklama yapılırken, ilgili satırda bulunan kodlar izah edilirken kullanılacaktır. Bu programın bilgisayarda `ilk.c` adı ile kaydedilmiştir.

Program 1.1: *Derlendikten sonra ekrana 'Merhaba Dünya!' yazar*

```
01: /* ilk.c: ilk C programi */
02: #include <stdio.h>
03:
04: main()
05: {
06:    printf("Merhaba Dünya!\n");
07: }
```

`/* ... */`

Programda, 1. satırda `/* ... */` sembolleri görülmektedir. Bu ifadeler arasında yazılan herhangi bir metin, işlem vb. satırlar, derleyici tarafından işlenmez (değerlendirilmez). Yani `/* */` ifadeleri açıklama operatörüdür.

NOT

Açıklama operatörü olarak C++ tarzı iki-bölü (//) de kullanılmaktadır. Günümüzde birçok C derleyicisi // operatörünü desteklemektedir. Bu operatörü kullanmadan önce derleyicinizin bu operatörü desteklediğinden emin olun.

```
 /*
    Bu satırlar derleyici tarafından
    değerlendirilmez.    Ayrıca    programın
C tarzı
    çalışma hızını da değiştirmez.
 */

 // Bu satırlar derleyici tarafından
 //   değerlendirilmez.   Ayrıca   programın
C++ tarzı
 // çalışma hızını da değiştirmez.
```

`#include <stdio.h>`

2. satırdaki `#include` deyimi, programda eklenecek olan başlık dosyanını işaret eder. Bu örnekte verilen başlık dosyası (header file) `stdio.h` dir. `#include <stdio.h>` ifadesi`stdio.h` dosyasının derleme işlemine dahil edileceğini anlatır[1]-[2]. Bu dosyalardan, Bölüm 20'de tekrar bahsedilecektir.

`main()`

4. satırdaki `main()` özel bir fonksiyondur. Ana program bu dosyada saklanıyor anlamındadır. Programın yürütülmesine bu fonksiyondan başlanır. Dolayısıyla her C programında bir tane `main()` adlı fonksiyon olmalıdır.

`printf()`

6. satırdaki `printf()` standart kütüphane bulunan ekrana formatlı bilgi yazdırma fonksiyondur. `stdio.h` dosyası bu fonksiyonu kullanmak için program başına ilave edilmiştir. Aşağıda `printf()` fonksiyonunun basit kullanımı gösterilmiştir.

Örnek kullanım şekli	***Ekranda yazılacak ifade***
`printf("Element: Aluminyum");`	`Element: Aluminyum`
`printf("Atom numarası = %d",13);`	`Atom numarası = 13`
`printf("Yoğunluk = %f g/cm3",2.7);`	`Yoğunluk = 2.7 g/cm3`
`printf("Erime noktası = %f derece",660.32);`	`Erime noktası = 660.32 derece`

Daha fazla bilgi için bkz. Bölüm 4.

1.4 Başlık Dosyaları

C dilinde bir program yazılırken, başlık dosyası (header file) olarak adlandırılan bir takım dosyalar `#include` önişlemcisi kullanılarak program içine dahil edilir. C kütüphanesinde bulunan birçok fonksiyon, başlık dosyaları içindeki bazı bildirimleri kullanır. Bu tür dosyaların uzantısı `.h` dir. ANSI C'deki standart başlık dosyaları şunlardır:

```
assert.h    locale.h    stddef.h
ctype.h     math.h      stdio.h
errno.h     setjmp.h    stdlib.h
float.h     signal.h    string.h
limits.h    stdarg.h    time.h
```

Bir çok C derleyicisinde yukarıdakilere ek olarak tanımlanmış başlık dosyaları da vardır. Bunlar derleyicinin yardım kısmından veya derleyicinin kullanım kılavuzundan öğrenilebilir. Ayrıca Bkz. Bölüm 20

`ilk.c` programında kullanılan başlık dosyası `stdio.h`, `#include <stdio.h>` ifadesi ile derleme işlemine dahil edilmiştir. `stdio.h` standard giriş/çıkış (STandarD-Input-Output) kütüphane fonksiyonları için bazı bildirimleri barındıran bir dosyasıdır. Programda kullanılan `printf()` fonksiyonunu kullanmadan önce bu başlık dosyası programın başına mutlaka ilave edilmelidir. Aksi halde derleme esnasında

```
undefined reference to _printf
```

şeklinde bir hata mesajı ile karşılaşılır.

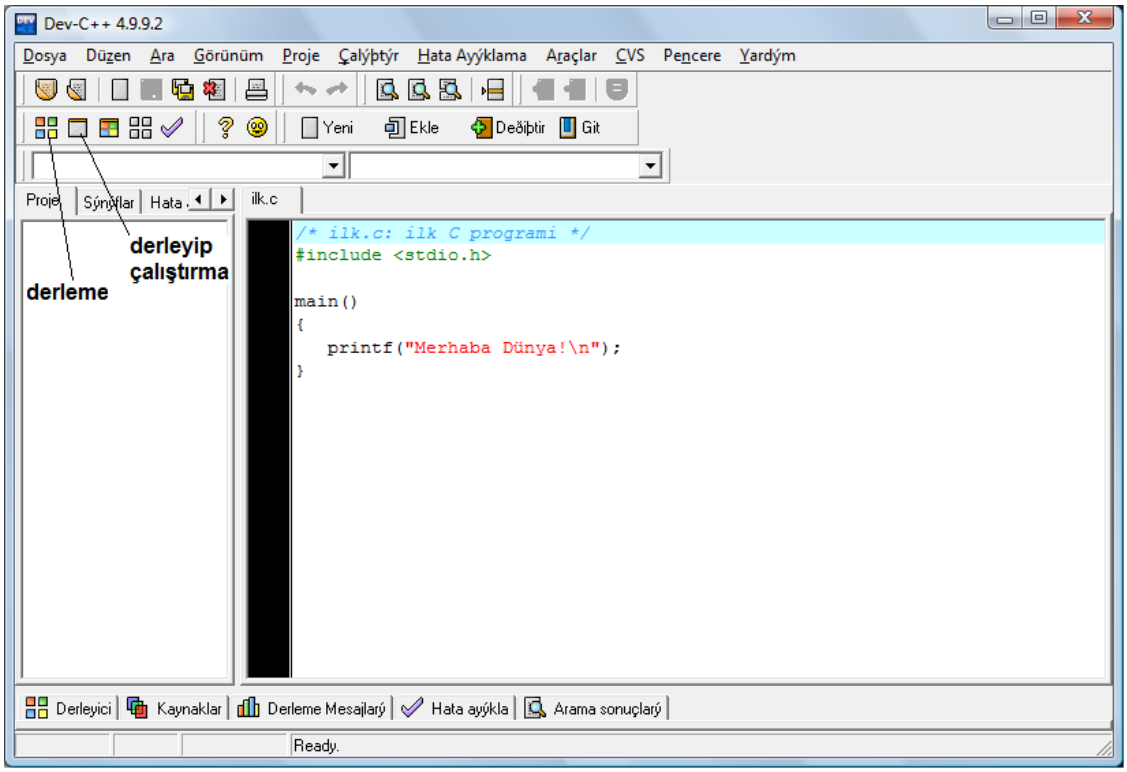
1.5 Kaynak Kodunun Derlenmesi

C programları veya kaynak kodları (source code) uzantısı `.c` olan dosyalarda saklanır. Kaynak kod, bir C derleyicisi (C compiler) ile nesne koduna (object code) daha sonra uygun bir bağlayıcı (linker) programı ile işletim sisteminde çalıştırılabilen (executable) bir koda dönüştürülür. Derleme işlemi ayrıntılı olarak Bölüm 22'de anlatılmıştır. Bazı işletim sistemleri ile kullanılan C Derleyicileri ve bu derleyicilerde `ilk.c` programının komut satırında nasıl derleneceği Tablo 1.1'de verilmiştir. Eğer ismi geçen derleyicinin bir editörü varsa `ilk.c` bu editör de derlenebilir.

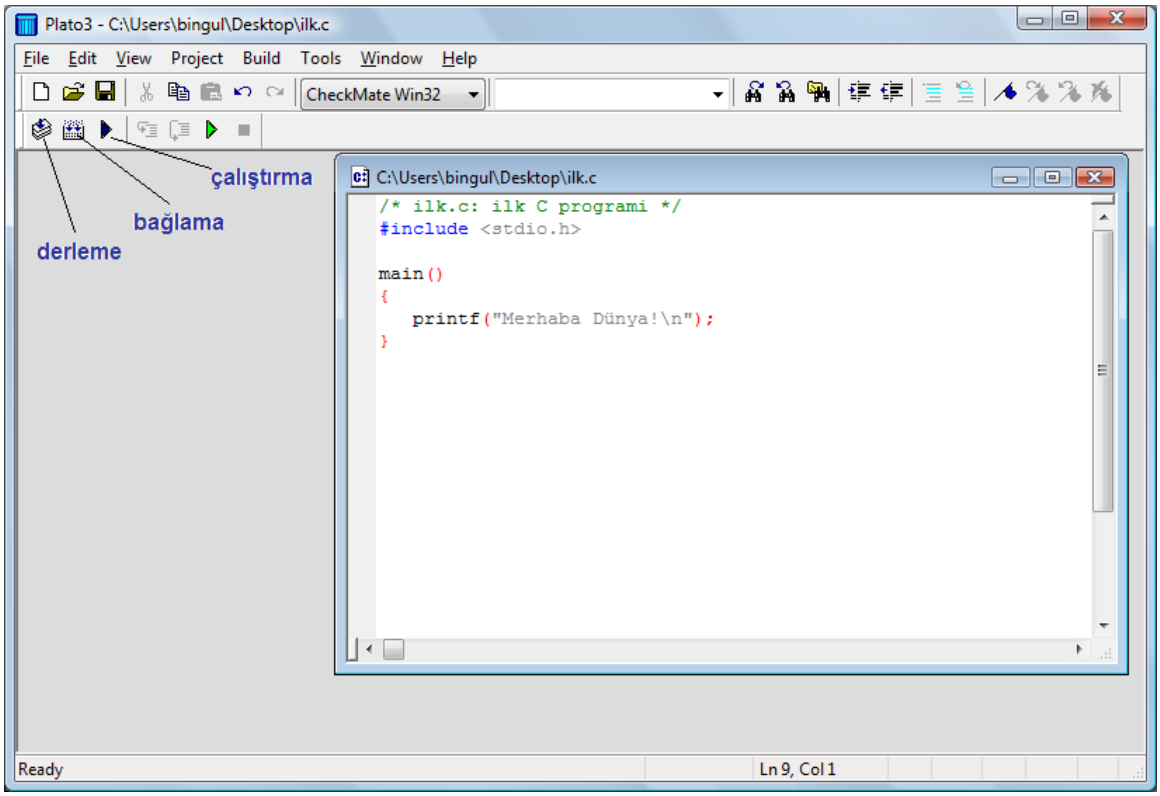
Tablo 1.1: *İşletim sistemleri, bazı derleyiciler ve derleme komutları*

İşletim Sistemi	Derleyici	Derleme	Çalıştırma
MS-DOS / Windows	Microsoft C	`cl ilk.c`	`ilk.exe`
	Borland Turbo C Web	`tcc ilk.c`	`ilk.exe`
	Borland C	`bcc ilk.c`	`ilk.exe`
	Zortec C	`ztc ilk.c`	`ilk.exe`
	GCC (GNU Compiler Collection) Windows için Web	`gcc ilk.c -o ilk.exe`	`ilk.exe`
UNIX / Linux	GCC (GNU Compiler Collection) Web	`gcc ilk.c -o ilk`	`./ilk` veya `nice ilk`

Bunların dışında, komut satırını kullanmadan, kodlarınızı Windows ortamında çalışan GCC tabanlı DevC++ veya Salford Plato3 derleyicileri ile derlemek mümkün. Bu tip derleyicilerde hata ayıklama işlemini kolaylaştırmak için kodlar farlı renkte gösterilir. Fakat program çıktıları için kullanılan ekran klasik DOS ekranıdır. Şekil 1.1 ve 1.2"de bu programların ekran görüntüleri verilmiştir.

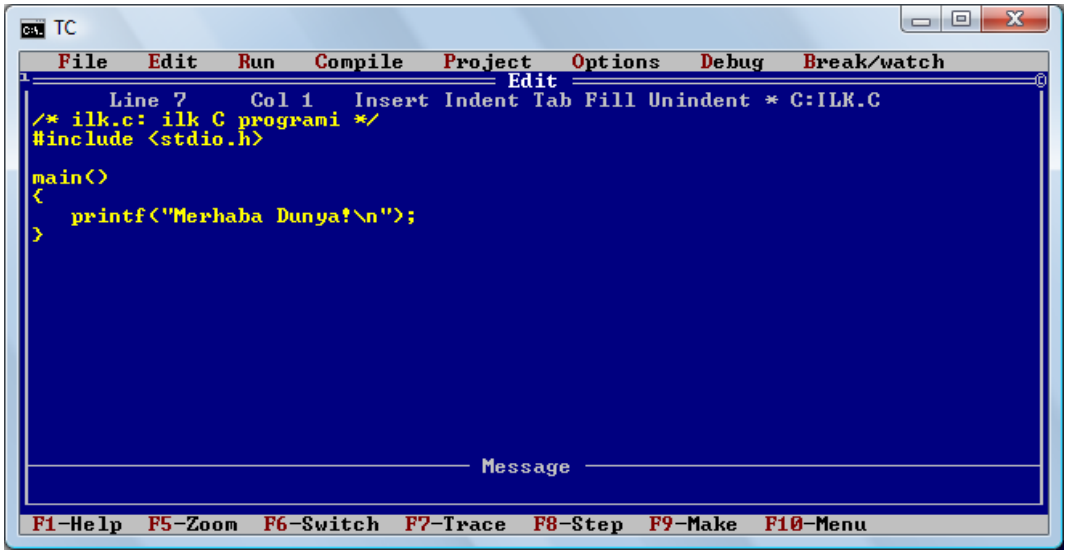


Şekil 1.1: *DevC++ derleyicine ait editör. Derleme ve çalıştırma işlemleri araç çubuğu üzerindeki butonlarla yapılır.*



Şekil 1.2: *Silverfrost Salford (Plato3) derleyicine ait editör. Derleme, bağlama ve çalıştırma işlemleri araç çubuğu üzerindeki butonlarla yapılır.*

Derslerimizde kullanılan kaynak kodları, Turbo C ve GCC derleyicileri ile komutsatırında derlenmiştir. Turbo C derleyicisi isteğe bağlı editörden veya komut satırından derlenebilir. Editörü başlatmak için C:\TC> dizini altındaki TC.EXE dosyasının çalıştırılması yeterlidir. Şekil 1.3'de Turbo C editör ortamı gösterilmiştir.



Şekil 1.3: *Turbo C derleyicisine ait editör. Derleme için F9, Derleme bağlama ve çalıştırma işlemleri için CTRL+F9 tuş kombinasyonu kullanılabilir..*

NOT
DevC++, Salford, GCC ve Turbo C derleyicilerini C/C++ Derleyicileri kısmında bulabilirsiniz.

`ilk.c` nin Borland Turbo C ve GCC Programları ile derlenmesi ve çalıştırılması:

DERLEME ve ÇALIŞTIRMA

MS DOS (Turbo C)	Linux (GCC)
`C:\TC> tcc ilk.c` `C:\TC> ilk.exe`	`$ gcc ilk.c -o ilk` `$ ./ilk`

`ilk.c` nin çıktısı:

ÇIKTI

```
Merhaba Dünya!
```

1.6 C Kodlarının Temel Özellikleri

Bir C programı aşağıda verilen özellikleri mutlaka taşımalıdır.

- Yazılımda kullanılacak olan her fonksiyon için ilgili başlık dosyası programın başına ileve edilmedlidir.
- Her C programı `main()` fonksiyonunu içermelidir.
- Program içinde kullanılacak olan değişkenler ve sabitler mutlaka tanımlanmalıdır.
- Satırın sonuna `;` işareti konmalıdır.
- Her bloğun ve fonksiyonun başlangıcı ve bitişi sırasıyla `{` ve `}` sembolleridir.
- C dilinde yazılan kodlarda küçük-büyük harf ayrımı vardır (case sensitive). Örneğin `A` ile `a` derleyici tarafından farklı değerlendirilir.
- Açıklama operatörü `/* */` sembolleridir.

1.7 Kod Yazımı için Bazı Tavsiyeler

- Program açıklamalarını ve döküman hazırlama işini program yazıldıkça yapın! Bu unutulmaması gereken çok önemli husustur.
- Değişken, sabit ve fonksiyon adları anlamlı kelimelerden seçilip yeterince uzun olmalıdır. Eğer bu isimler bir kaç kelimeden oluşacak ise, kelimeler alt çizgi (_) ile ayrılmalıdır veya her kelime büyük harfle başlamalıdır. Örneğin:

```
int son_alinan_bit;
void KesmeSayisi();
float OrtalamaDeger = 12.7786;
```

- Sabitlerin bütün harflerini büyük harfle yazın. Örneğin:

```
#define PI 3.14;
const int STATUS = 0x0379;
```

- Her alt yapıya girerken birkaç boşluk veya TAB tuşunu kullanın. Bu okunabilirliği arttıracaktır. Örneğin:

```
k = 0;
for(i=0; i<10; i++)
{
      for(j=0; j<i; j+=2)
      {
            do{
                  if( j>1 )  k = i+j;

                  x[k] = 1.0/k;
              }while(k!=0);
      }
}
```

- Aritmetik operatörler ve atama operatörlerinden önce ve sonra boşluk karakteri kullanın. Bu, yazılan matematiksel ifadelerin daha iyi anlaşılmasını sağlayacaktır.Örneğin:

```
h_max = pow(Vo,2) / (2*g);
Tf    = 2*Vo/g;
Vy    = Vo - g*t;
y     = Vo*t - (g*t*t)/2.0;
z     = ( a*cos(x) + b*sin(x) )*log( fabs(y) );
```

- Program bittikten sonra tekrar tekrar programınızı inceleyerek, programınızı daha iyi şekilde yazma yollarını arayın ve aynı fonksiyonları daha kısa algoritmalarla ve/veya daha modüler şekilde elde etmeye çalışın.

Ders 2: *Veri Tipleri, Değişkenler ve Sabitler*

Giriş

Orta ve yüksek seviyeli dillerin hemen hemen hepsinde veri tipi ve değişken kavramı bulunmaktadır. Bu kısımda C programlama dilindeki temel veri tipleri, tanımlayıcılar, değişkenler ve sabitler konu edilecektir.

2.1 Veri Tipleri

Veri tipi (data type) program içinde kullanılacak değişken, sabit, fonksiyon isimleri gibi tanımlayıcıların tipini, yani bellekte ayrılacak bölgenin büyüklüğünü, belirlemek için kullanılır. Bir programcı, bir programlama dilinde ilk olarak öğrenmesi gereken, o dile ait veri tipleridir. Çünkü bu, programcının kullanacağı değişkenlerin ve sabitlerin sınırlarını belirler. C programlama dilinde dört tane temel veri tipi bulunmaktadır. Bunlar:

```
char
int
float
double
```

Fakat bazı özel niteleyiciler vardır ki bunlar yukarıdaki temel tiplerin önüne gelerek onların türevlerini oluşturur. Bunlar:

```
short
long
unsigned
```

Bu niteleyiciler sayesinde değişkenin bellekte kaplayacağı alan isteğe göre değiştirilebilir. Kısa (`short`), uzun (`long`), ve normal (`int`) tamsayı arasında yalnızca uzunluk farkı vardır. Eğer normal tamsayı 32 bit (4 bayt) ise uzun tamsayı 64 bit (8 bayt) uzunluğunda ve kısa tamsayı 16 biti (2 bayt) geçmeyecek uzunluktadır. İşaretsiz (`unsigned`) ön eki kullanıldığı taktirde, veri tipi ile saklanacak değerin sıfır ve sıfırdan büyük olması sağlanır. İşaretli ve işaretsiz verilerin bellekteki uzunlukları aynıdır. Fakat, işaretsiz tipindeki verilerin üst limiti, işaretlinin iki katıdır.

NOT
Kısa ve uzun tamsayı tutacak tanımlayıcılar
için `int` anahtar kelimesinin yazılmasına gerek yoktur.

```
short s;   /* short int s; anlamında */
long k;    /* long int k;  anlamında */
```

Bir C programı içerisinde, veri tiplerinin bellekte kapladığı alan `sizeof` operatörü ile öğrenilebilir. İlgi cekici olan, bu alanların derleyiciye ve işletim sistemine bağlı olarak değişiklik göstermesidir. Program 2.1'de, `sizeof` operatörü kullanılarak, veri tiplerinin bellek uzunlularının nasıl ekrana yazdırılacağı gösterilmiştir. Programın çıktısı, farklı derleyiciler ve işletim sisteminde denendiğinde bu durum daha iyi anlaşılır. Lütfen inceleyin.

Program 2.1: *Değişken tipleri ve türevlerinin bellekte kapladıkları alanlar*

```
01: /* 02prg01.c : sizeof operatörünün kullanımı */
02:
03: #include <stdio.h>
04:
05: main()
06: {
07:   printf( "char           : %d bayt\n",
08: sizeof(char));
09:   printf( "short          : %d bayt\n",
10: sizeof(short));
11:   printf( "int            : %d bayt\n",
12: sizeof(int));
13:   printf( "long           : %d bayt\n",
14: sizeof(long));
15:   printf( "unsigned char  : %d bayt\n",
16: sizeof(unsigned char));
17:   printf( "unsigned short : %d bayt\n",
18: sizeof(unsigned short));
      printf( "unsigned int   : %d bayt\n",
    sizeof(unsigned int));
      printf( "unsigned long  : %d bayt\n",
    sizeof(unsigned long));
      printf( "float          : %d bayt\n",
    sizeof(float));
      printf( "double         : %d bayt\n",
    sizeof(double));
      printf( "long double    : %d bayt\n",
    sizeof(long double));
    }
```

ÇIKTI

Windows (32 bit) Turbo C	Windows (32 bit) Salford	Linux (32 bit) GCC	Linux (64 bit) GCC
char : 1 bayt short : 2 bayt int : 2 bayt long : 4 bayt unsigned char : 1 bayt unsigned short : 2 bayt unsigned int : 2 bayt unsigned long : 4 bayt float : 4 bayt double : 8 bayt long double : 10 bayt	char : 1 bayt short : 2 bayt int : 4 bayt long : 4 bayt unsigned char : 1 bayt unsigned short : 2 bayt unsigned int : 4 bayt unsigned long : 4 bayt float : 4 bayt double : 8 bayt long double : 10 bayt	char : 1 bayt short : 2 bayt int : 4 bayt long : 4 bayt unsigned char : 1 bayt unsigned short : 2 bayt unsigned int : 4 bayt unsigned long : 4 bayt float : 4 bayt double : 8 bayt long double : 12 bayt	char : 1 bayt short : 2 bayt int : 4 bayt long : 8 bayt unsigned char : 1 bayt unsigned short : 2 bayt unsigned int : 4 bayt unsigned long : 8 bayt float : 4 bayt double : 8 bayt long double : 16 bayt

`int` veritipi ve türevleri ile hesaplanabilecek en küçük ve en büyük tamsayılar için aşağıdaki formül kullanılabilir:

Alt sınır = $-2^{8*sizeof(tip)}$
Üst sınır = $2^{8*sizeof(tip)}-1$

Örneğin 4 baytlık bir `int` tipi için:
Alt sınır = $-2^{8*sizeof(int)} = -2^{32} = -2147483648$
Üst sınır = $2^{8*sizeof(int)}-1 = 2^{32}-1 = 2147483647$

Tablo 2.1'de bütün tipler, bellekte kapladıkları alanlar ve hesaplanabilcek (bellekte doğru olarak saklanabilecek) en büyük ve en küçük sayılar listelenmiştir.

Tablo 2.1: *Değişken tipleri ve bellekte kapladıkları alanlar*

Veri Tipi	Açıklama	Bellekte işgal ettiği boyut (bayt)	Alt sınır	Üst sınır
`char`	Tek bir karakter veya küçük tamsayı için	1	-128	127
`unsigned char`			0	255
`short int`	Kısa tamsayı için	2	-32,768	32,767
`unsigned short int`			0	65,535
`int`	Tamsayı için	4	-2,147,483,648	2,147,483,647
`unsigned int`			0	4,294,967,295
`long int`	Uzun tamsayı için	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
`unsigned long int`			0	18,446,744,073,709,551,615
`float`	Tek duyarlı gerçel sayı için (7 basamak)	4	-3.4e +/- 38	+3.4e +/- 38
`double`	Çift duyarlı gerçel sayı için (15 basamak)	8	-1.7e +/- 308	+1.7e +/- 308

2.2 Değişkenler

Değişkenler bilgisayarın geçici belleğinde bilginin saklandığı gözlere verilen sembolik adlardır. Bir C programında, bir değişken tanımlandığında bu değişken için bellekte bir yer ayrılır. Her değişkenin tuttuğu değerin nasıl bir veri olduğunu gösteren (önceki bölümde anlatılan) bir veri tipi vardır [1], [3].

C programlama dilinde, değişkenler ve sabitler programın başında bulunmalıdır. Bazı uygulamalarda değişkenin bir başlangıç değerinin olması istenir. Böyle durumlarda değişken bildirilirken başlangıç değeri verilebilir. Örneğin:

```
char isim='X', z;     /* değer atamak zorunlu değil */
int  sayi=0, n;
float toplam=0.0, sonuc=22.14;
```

Değişken isimleri verirken bazı kurallara uymak zorunludur. Bunlar:

- Değişken adları en fazla 32 karakterden oluşabilir. 32 karakterden uzun değişken adları ilk 32 karakteri değerlendirilir. Geriye kalan karakterler işleme tabi tutulmaz.
- Değişken adları ingiliz alfabesinde bulunan karakterler (A-Z) veya (a-z) yada rakamlar (0-9) ile yazılmalıdır. Türkçe karakterler, özel karakter veya boşluk karakteri kullanılamaz.
- Değişken adları herhangi bir rakam ile başlayamaz. Ilk karakter bir harf olamalıdır. Sonrakiler rakamlardan oluşabilir.
- Aşağıda verilen kelimeler ANSI C 'nin anahtar kelimeleridir (key words) ve değişken ismi olarak kullanılamaz.
-

```
auto        double    int         struct
break       else      long        switch
case        enum      register    typedef
char        extern    return      union
const       float     short       unsigned
continue    for       signed      void
default     goto      sizeof      volatile
do          if        static      while
```

Bu kurallara göre aşağadaki değişken (sabit, fonksiyon) adlarının geçerliliğini inceleyiniz.

Değişken/Sabit/Fonksiyon/Yapı Adı	**Geçerlilik**	**Açıklama**
`asal`	geçerli	-
`Momentum`	geçerli	-
`ivme`	geçerli	-
`olasilik`	geçerli	-
`IsikHizi`	geçerli	-
`isik_hizi`	geçerli	Alt çizgi karakteri '_' kullanılabilir
`isik hizi`	geçersiz	Boşluk karakteri kullanılamaz
`ışık_hızı`	geçersiz	Türkçe karakter kullanılamaz
`1Bit`	geçersiz	rakam ile başlanamaz
`typedef`	geçersiz	Anahtar kelimelerden birisi kullanılamaz

2.3 Sabitler

Sabit bildirimi, başlangıç değeri verilen değişken bildirimi gibi yapılır. Ancak, veri tipinin önüne `const` anahtar sözcüğü konmalıdır. Örneğin:

```
const float   PI = 3.142857;
const double  NOT= 12345.8596235489;
const int     EOF= -1;
const char[] = "devam etmek için bir tuşa basın...";
```

gibi sabit bildirimleri geçerli olup bunların içerikleri program boyunca değiştirilemez. Yalnızca kullanılabilir. Genellikle, sabit olarak bildirilen değişken isimleri büyük harflerle, diğer değişken isimlerinin ise küçük harflerle yazılması (gösterilmesi) C programcıları tarafından geleneksel hale gelmiştir.

Birçok C programında sabitler `#define` önişlemci komutu ile de tanımlandığını görebilirsiniz. Bu komutla sabit bildirimi, bir program parçasına ve makro fonksiyon tanımlaması yapılabilir. Bir program geliştirilirken simgesel sabitlerin kullanılması programın okunurluğunu arttırır ve bazen gerekli de olabilir. Aşağıda verilen simgesel sabit bildirimleri geçerlidir.`#define` önişlemcisi ile makro fonksiyon tanımalama işlemi, Bölüm 8 ve Bölüm 20'de anlatılacaktır.

```
#define MAX     100
#define DATA    0x0378
#define YARICAP 14.22
```

2.4 Rakamsal Bilgiler

C programlama dili içinde tanımlanabilecek sabit rakamlar *rakamsal bilgi* (literal) olarak adlandırılır. Her veri tipi kendi rakamsal bilgisine sahiptir. Bu bilgiler, kaynak kod içerisinde, özel değerleri ifade eder. Örneğin aşağıdaki atama işleminde 25 ve 17.2 sayıları gibi:

```
i = 25;    /* 25, int tipinde bir rakamsal bilgidir */
r = 17.2;  /* 17.2, double tipinde bir rakamsal bilgidir */
```

C dilinde bütün tamsayı sabitler varsayılan (default) olarak `int` tipinde, gerçel sayı sabitler varsayılan olarak `double` tipindedir. Ancak sabitleri gösteren rakamların sonuna eklenecek U (veya u), L (veya l) ve F (veya f) harfleri ile bu durum değiştirilebilir. Bu yüzden, aşağıdaki atamalar aynı anlamda değildir.

```
i = 25;    /* int           rakam */
i = 25U;   /* unsigned int  rakam */
i = 25L;   /* long int      rakam */
i = 25UL;  /* unsigned long rakam */
i = 25L;   /* long int      rakam */

r = 17.2;  /* double        rakam */
r = 17.2L; /* long double   rakam */
r = 17.2F; /* float         rakam */
```

Tamsayı (`int`) rakamsal bilgiler, 8 (oktal) ve 16 (hexadesimal) sayı tabanında da gösterilebilir. Bunun için sabit rakamın başına, 8 tabanı için 0 (sıfır) ve 16 tabanını için 0x sembolleri eklenir. 16'lık sistemdeki hafler büyük (A, B, C, D, E ve F) veya küçük (a, b, c ,d, e ve f) olabilir. Buna gösterime göre, aşağıdaki atmalar aynı anlamadadır:

```
i = 75;    /* i = 75, 10 tabanında */
i = 0113;  /* i = 75,  8 tabanında */
i = 0x4b;  /* i = 75, 16 tabanında */
i = 0x4B;  /* i = 75, 16 tabanında */
```

Gerçel sayılar *ondalıklı* veya *üstel* olmak üzere iki biçimde gösterilebilir. Örneğin 123.456 sayısının aynı anlama gelen dört farklı gösterimi aşağıda verilmiştir. Üstel gösterimde, 1.23456e+2 veya 1.23456E+2 sayısı matematikteki 1.23456 x 10^2 gösterimi ile eşdeğerdir.

```
x = 123.456;      /* ondalıklı gösterimi */
x = 123.456e+0;   /* üstel gösterim */
x = 1.23456e+2;   /* üstel gösterim */
x = 1234.56E-1;   /* üstel gösterim */
```

Karakter sabitler, bir harf için tek tırnak, birden çok karakter için çift tırnak içinde belirtilirler.

```
'A'               /* bir karakter */
"Merhaba Dunya"   /* bir karakter kümesi */
```

Program 2.1'de, program içinde tanımlanan değişken sabitlerin ekrana nasıl yazdırılacağı gösterilmiştir.

Program 2.2: *Değişkenlerin ve sabitlerin ekrana yazdırılması*

```
01: /* 02prg02.c : Değişkenler ve sabitlerin ekrana
02: yazdırılması */
03:
04: #include <stdio.h>
05:
06: #define PI 3.141593
07:
08: int main()
09: {
10:   const int MAX = 100;
11:   char   c = 'a';
12:   char  *s = "Bu bir sicim";
13:   int    i = 22;
14:   float  f = 33.3;
15:   double d = 44.4;
16:
17:   printf("PI = %lf\n",PI);
18:   printf("MAX= %d\n", MAX);
19:   printf("c  = %c\n", c);
20:   printf("s  = %s\n", s);
21:   printf("i  = %d\n", i);
22:   printf("f  = %f\n", f);
23:   printf("d  = %lf\n",d);
24:
25:  return 0;
    }
```

ÇIKTI

```
PI = 3.141593
MAX= 100
c  = a
s  = Bu bir sicim
i  = 22
f  = 33.299999
d  = 44.400000
```

2.5 Değişken Bildirim Yerleri ve Türleri

Yerel (local) Bildirim

Yerel değişkenler kullanıldığı fonksiyon içerisinde bildirilir. Yalnızca bildirildiği fonksiyon içerisinde tanınır ve kullanılabilir.

```
 int topla(int a,int b)
 {
 /* yerel (local) değişken c nin bildirimi */
  int c;
  c = a + b;
  return c;
}
```

Genel (general) Bildirim

Genel değişkenler bütün fonksiyonların dışında bildirilir. Bir değişken program boyunca sürekli olarak kullanılıyorsa genel olarak bildirilmelidir.

```
#include <stdio.h>

void karesi();

/*  m ve n global tip değişkendir.
    Bu iki değişken tüm program boyunca kullanılmaktadır. */

int m,n;

main()
{
  m=7;
  karesi();
  printf("%d nin karesi %d dir",m,n);
}

void karesi(){
  n = m*m;
}
```

2.6 Tip Dönüşümleri

Bir formül içerisinde bir çok değişken veya sabit olabilir. Bu değişken ve sabitler birbirinden farklı tipte olursa, hesap sonucunun hangi tipte olacağı önemlidir. Bir bağıntıda, içeriği dönüşüme uğrayan değişkenler eski içeriklerini korurlar. Dönüştürme işlemi için geçiçi bellek alanı kullanılır; dönüştürülen değer kullanıldıktan sonra o alan serbest bırakılır.

```
char kr;
int tam;
long int ltam;
unsigned int utam;
short int stam;
float f;
double d;
```

bildirimlerine göre:

```
Bağıntı       Sonuç Tipi
-------       ----------
kr+5          int
kr+5.0        double
d+tam         double
f+d-2         double
utam-tam      unsigned
ltam*tam      long
tam/2         int
tam/2.0       double
```

> **NOT**
> Tamsayılar arası bölme kesme hatalarına (truncation error) neden olur.
> Bunun anlamı *iki tamsayının oranı yine bir tamsayıdır*.
> örneğin: 4/2=2; ama 3/2=1 (1.5 değil).

Bir değişkenin sabit değerin veya bağıntının önüne tür veya takı (cast) yazılarak sonucun hangi tip çıkması istendiği söylenebilir. Genel yazım biçimi:

```
(tür tipi)  bağıntı;
```

Örneğin:

```
int x=9;
float a,b,c;
double d;
...
a = x/4;
b = x/4.0;
c = (float) x/4;
```

işleminin sonucunda `a` değişkenine 2.0, `b` ve `c` değişkenlerine 2.25 değeri aktarılır. Yani `9/4` ile `9/4.0` farklı anlamdadır.