2. BÖLÜM (2nci hafta)

Nesne merkezli programlamada (object oriented programming), program içinde kullanılacak nesne verilerinin belirlenmesi, yapılacak işlemler ve bu nesneye yönelik algoritmanın ve kodların yazılması gerekmektedir. Bilgisayar programlamada nesne ekrana bir mesaj yazdırma olabilir, yazıcıya doküman yollanması olabilir, kayıt ortamına verilerin yazdırılması veya okutulması olabilir. Özetlemek gerekirse bilgisayar programlamada yapılmak istenen iş nesne (hedef) olarak tanımlanabilir. C++ da nesneye yönelik program yazarken programda kullanılacak tiplerin (types), değişkenlerin (variables), fonksiyonların (functions) ve diğer niceliklerin önceden tanımlanması yani programa bildirilmesi (definition) gerekmektedir. Bu açıdan C++ programlama dili hiyerarşik (sıralı) bir yapıya sahiptir. Herşey derleyicinin anlayacağı şekilde tanımlanmalıdır veya bildirilmelidir. Bilgisayar, işleyeceği verileri (data) ancak çeşitli simgeler aracılığı ile belleğinde tutar. Bu kesimde C++ da kullanılabilen veri tipleri, simgeler ve bunlarla ilgili olarak yazım kuralları ile ilgili temel bilgiler verilecektir.

2.1 VERİ TİPLERİ (Data Types)

C++ programlama dilinde kullanılan veri tipleri temelde üç gruba ayrılır: Tamsayılar (integers), gerçel sayılar (real numbers) ve karakterler (characters). Çizelge 2-1 de temel veri tipleri, yazımı, bellekte işgal ettikleri alanlar, bellekte saklayabilecekleri en küçük ve en yüksek değerleri verilmektedir. Verileri bilgisayar belleğinde saklayabilmek ve daha sonra kullanabilmek için çeşitli simgelerin kullanılması gerektiğini belirtmiştik. Bu amaçla bu simgeler isimlendirilir. Bu simgeler topluluklarına değişkenler veya değişmezler şeklinde isimler verilir. Değişkenler, programın çalışması esnasında sakladıkları verilerin değişmesine izin verirlerken, değişmezlerde saklanan veriler programın çalışması esnasında değiştirilemezdirler.

Çizelge 2-1 C++ da kullanılan veri tipleri.

Veri Tipleri	Yazılışı	Kapladığı Alan	En Düşük Değeri	En Yüksek Değeri
Tam sayılar	unsigned int (+)	2 byte (2×8=16 bit)	0	$2^{2\times8}-1=65535$
	int (– ve +)	2 byte (2×8=16 bit)	$-2^{2\times8-1}=-32768$	$2^{2\times8-1}-1=32767$
	unsigned long int	4 byte (4×8=32 bit)	0	$2^{4\times8}-1=4,294,967,295$
	long int	4 byte (4×8=32 bit)	$-2^{4\times8-1}=-2,147,483,648$	$2^{4\times8-1}-1=2,147,483,647$
Gerçel sayılar	float (+ ve –)	4 byte (4×8=32 bit) 23 bit kesirsel kısım $=1/2^{23}=1.192\times10^{-7}$, 7 bit üstel kısım=$2^7$=128 ise $2^{128}\approx 10^{38}$	$-2^{-128}=-1.7\times10^{-39}$	$2^{128-1}=1.7\times10^{38}$
	double(+ ve –)	8 byte (8×8=64 bit) 52 bit kesirsel kısım $=1/2^{52}\approx10^{-15}$, 10 bit üstel kısım=2^{10}=1024 ise $2^{1024}\approx10^{308}$	-1.176×10^{-308}	3.403×10^{308}
Karakter	Char	1 byte		

Çizelge 2-2 C++ dilinde değişmez ve değişkenlerde kullanılabilecek karakterler.

Alfanümerik karakterler	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
Rakamlar	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Alt çizgi	_
Özel Karakterler	" , ! . ; : ' ( ) [ ] { } | / \ ~ + # % & ^ * - = <

Çizelge 2-2 de C++ programlama dilinde değişkenlerin veya değişmezlerin isimlendirilmesinde kullanılabilecek simgeler verilmektedir.

2.2 DEĞİŞKEN veya DEĞİŞMEZLERİN İSİMLENDİRME KURALLARI

C++ da değişken veya değişmezleri isimlendirilmesinde, ingiliz alfabesinin küçük ve büyük harfleri, arab rakamları ve alt çizgi (_) karakteri kullanılır. İsimlendirmede

- özel karakterler ve türkçe karakterler **ÜüÇçĞğİıÖöŞş** (veya başka dillerin özel karakterleri) kullanılmamalıdır.
- C++ da, küçük harfler ile büyük harfler birbirinden **farklı** anlam taşırlar,
- İsmin ilk karakteri bir harf veya _ altçizgi olmalıdır. İlk karakter sayı olamaz.
- Değişken ismi 32 karakteri aşamaz.
- C++ için ayrılmış sözcükler (reserved words),
- Programcı tarafından kullanılmış bir isim tekrar tanımlanarak kullanılamaz.

Bir C++ programı, içinde kullanılan değişkenlerdeki karakterlerin büyük veya küçük harfle kullanılmasına karşı duyarlıdır. Örnek olarak, `KUTLE`, `kutle`, `KuTLe` ile `Kutle` isimlendirmeleri bellekte birbirinden farklı bölgeleri (değişkenleri) tanımlar. Bir C++ programında program içinde kullanılan değişkenler uzun isimlerle yazılarak, programcıya bu isimlendirmenin ne amaçla kullanılacağı (bazı şeyleri hatırlatması) olanağı sağlanır. Örnek olarak `PI`, `AVAGADRO_SAYISI` ve `ISIK_HIZI`, `DalgaBoyu` (sabitler için büyük harfler seçilmesi programcılık açısından yararlı olabilir yada kendiniz bu tür işlemler için kural oluşturunuz) veya `Goreli_Kutle`, `Potansiyel_Enerji`, `Personel_Adi` (değişkenler için küçük karakterler kullanılması programcılık açısından yararlı olabilir) verilebilir. Bu isimlendirmeler programcılıkta literal olarak adlandırılır. Ayrıca hazırlayacağınız program, sistemin belleğini iyi bir şekilde kullanmalıdır. Yanlış bellek kullanımlarından sistem etkilenebilir veya sistem kilitlenenebilir.

Örnek 2-1 Aşağıda değişkenler kullanım amaçlarına göre tanımlanmış ve isimlendirilmiştir.

```
int SinifSayisi=20;//okuldaki sınıfların sayısı tamsayı değişkenidir
int Yas;//bir insanın yaşı tamsayı değişken olarak tanımlanmıştır
```

```
int Faktoriyel;//bir sayının faktöriyeli tamsayı değişken
int const y=5;//tamsayı değişmez olarak tanımlanmıştır
float KDV=0.18;//katma değer vergisi tek duyarlı değişken
float const PI=3.1415926;//π-tek duyarlı sabit tanımlanmıştır
double const e=2.718281;//çift duyarlı sayı sabit tanımlanmıştır
char Adi[20];//20 elemanlı karakter değişkeni olarak tanımlanmıştır
char UlkeKodu[2]={'T','R'}; // 2 elemanlı karakter değişkeni
char mesaj[1]="Dikkat! Hatalı değer giriliyor." ;
char const SehirAdi[ ] = "Ankara";
```

Değişken olarak tanımlanan nicelikler programın çalışması esnasında bellekte sakladıkları değerlerin değişmesine olanak verirler. Yukarıdaki Örnek 2-1 de yapılan tanımlamalarda bazı değişkenlerin sakladıkları değerler programın çalışması esnasında değişebilmekte veya değişmemektedir.

2.3 TAMSAYILAR, GERÇEL SAYILAR ve KARAKTERLER

C++ programlama dili kullanılarak yazılan programlarda verilerin 3 ana grubu ayrılarak değerlendirildiğini belirtmiştik: tamsayılar, gerçel sayılar ve karakterler. Bunlarda kullanım amacına göre küçük tamsayılar, büyük tamsayılar, tek duyarlıklı sayılar, çift duyarlıklı sayılar veya karakterler olarak sınıflandırılabilir.

Örnek 2-2 Aşağıdaki program d değişkeninde saklı bir karakterin ekrana yazdırılması işlemini *printf* komutu ile yapmaktadır. Programın ekran çıktısı sağdaki kutucuktaki gibidir.

```
#include <stdio.h>
#include <conio.h>
main (){
  char d;  d='x';
  printf("Karakter : %c", d);
getch();}
```

Yukarıdaki programın çalıştırılması sonucu ekran çıktısı aşağıdaki gibidir:

```
Karakter : x
```

Örnek 2-3 Aşağıdaki algoritma ve programda tamsayı ve karakter içerikli veriler klavyeden girilerek değişkenlere (bilgisayar belleğine) aktarılır ve sonra bu değişkenlerdeki veriler ekrana yazdırılır. Algoritma programın işleyişini özetlemektedir.

Algoritma 2-1 Karakter ve tamsayı değişkenlerinin tanımlanması.

```
1. Başla,
2. Tamsayı değişkenlerini tanımla, i1, i2, i3,
3. Karakter değişkenlerini tanımla, c1, c2, c3,
4. i1, i2, i3, c1, c2, c3 değişkenlerine değerler gir,
5. i1, i2, i3, c1, c2, c3 değişkenlerindeki değerleri yaz,
6. Son
```

Program 2-1 Karakter ve tamsayı değişkenlerin kullanılması.

```
#include <stdio.h>
#include <conio.h>
main (){
  char c1, c2, c3;
  int  i1, i2, i3;
  scanf ("%d %d %d %c %c %c", &i1, &i2, &i3, &c1, &c2, &c3);
  printf("%d %d %d %c %c %c", i1, i2, i3, c1, c2, c3);
  getch();}
```

Programda karakter ve tamsayı değişkenleri tanımlandıktan sonra bunlara veri girişinin (`scanf()`) klavyeden yapılacağı belirtilmiştir. Daha sonra değişkenlere aktarılan bilgiler ekrana (`printf()`) yazdırılır. C/C++ programlama dilinde aynı anda birden fazla karakter bilgisayar belleğine yani değişkene aktarılmaz, işlemler sırayla yapılır. Karakter verileri ancak teker teker değişkenlere aktarılır (bilgisayara okutulabilir). Tam sayı ve karakter değişkenlerinin önündeki & işareti verinin aktarılacağı bellek bölgesinin önceden temizleneceğini göstermektedir.

Programın çalıştırılması sonucu ekran çıktısı aşağıdaki gibidir:

```
3 4 5 a b c
3 4 5 a b c
```

Örnek 2-4 Aşağıda tek ve çift duyarlı sayılara örnek algoritma ve program verilmektedir. Veriler program içerisindeki satırlara yazılarak bilgisayar belleğine (yani değişkenlere) aktarılır ve sonra bu değişkenlerdeki veriler ekrana yazdırılır. Programın çalıştırılması sonucu ekran çıktısı aşağıda verilmektedir.

Algoritma 2-2 Tek ve çift duyarlı değişkenler.

```
1. Başla,
2. Tek duyarlı h değişkenini tanımla,
3. Tek duyarlı v=0.865 değişkenini tanımla ve değer aktar,
3. Çift duyarlı p değişkenini tanımla,
4. h=1.50,
5. p=13.4e-40,
6. h ve v değişkenlerindeki değerleri yaz,
7. p değişkenindeki değeri yaz,
8. Son.
```

Program 2-2 Tek ve çift duyarlı sayılar.

```
#include <stdio.h>
#include <conio.h>
main () {
  float h, v=0.865; double p;
```

```
  h=1.50; p=13.4e-40;
  printf("h=%1.3f  v=%1.3f \n", h, v);
  printf("p=%2.2e  \n", p);
  getch();}
```

Ekran çıktısı :

```
h=1.500  v=0.865
p=1.34e-039
```

2.4 ARİTMETİK İŞLEMCİLER

C/C++ programlama dilinde çeşitli aritmetik işlemleri (toplama, çıkarma, çarpma, bölme, vs.) yapabilmek için kullanılabilecek karakterler Çizelge 2-3 de verilmektedir. Bu işlemcilerden sonra veri bilgisayar belleğine aktarılacaksa mutlaka veri aktarma = (eşit) simgesi veya işlemcisi kullanılmalıdır. = (eşit) simgesi ile klavyeden veya başka bir ortamdan (örneğin magnetik veya optik) bilgisayar belleğine veri transferi yapılabilmektedir.

Çizelge 2-3 C++ programlama dilinde kullanılan aritmetik işlemciler.

İşlem	Aritmetik gösterimi	C++ da kullanımı
Toplama	2+3	2+3
Çıkartma	2−3	2−3
Çarpma	2×3	2*3
Bölme	$\frac{2}{3}$	2/3

2.4.1 Aritmetik İşlemlerde Öncelik Sırası

Bilgisayar ortamında (= simgesinin sağ tarafında) aritmetik işlemler belirli bir (hiyerarşik) sıraya göre yapılır. Öncelikle yapılması gereken işlemler varsa bunlar için parantezler kullanılmalı ve işlemleri doğru yaptırabilmek için mutlaka parantezlere dikkat edilmelidir. Aşağıda Çizelge 2-4 de çeşitli işlemcilerin öncelikleri listelenmektedir. Formülleri yazarken, ifadeler kullanılırken parantezlere, bölme işlemlerine özellikle dikkat edilmelidir. Yapılan mantıksal hataların bazen bulunması çok güç olmaktadır.

Çizelge 2-4 C++ programlama dilinde aritmetik işlemlerde öncelik sırası.

- Parantez içi öncelikle yapılır. İçiçe parantezler varsa en içteki parantez içindeki işlemler ilk önce yapılır, daha sonra işlemler dışa doğru açılarak yapılır.
- Aynı önceliğe sahip işlemcilerin (operatörlerin) aynı ifade içinde kullanılması durumunda, işlemler (eşitliğin sağ tarafında) soldan sağa doğru yapılır.
- ++ (artırım) veya – – (eksiltim) işlemleri öncelikle yapılır,
- – işaret değiştirme işlemi öncelikle yapılır,
- *, /, % işlemlerinde önce çarpma işlemi yapılır,
- +, – işlemlerinde önce toplama yapılır.

Örnek 2-5 `x1=(-b+sqrt(b*b-4*a*c))/(2*a)` ifadesinde öncelikli işlem pay kısmında en içte parantezle belirtilmiş olan `(b*b-4*a*c)` dedir. Önce buradaki işlemler yapılır ve buradada öncelik sırası en soldaki çarpmadadır `(b*b)`. Yani eşit öcelikli işlemler durumunda soldan sağa doğru gidilmelidir. Sağdaki `-4*a*c` işlemi için önce işaret değişimi sonra çarpma işlemleri ve en sonunda toplama/çıkarma `(b*b-4*a*c)` işlemi gerçekleştirilir. Toplama işleminden sonra elde edilen değerin karekökü hesaplanır. Pay ve paydadaki işlemler öncelikli olarak tamamlanıp en sonunda bölme işlemi yapılır.

Örnek 2-6 `P=m*g*h` işleminde öncelik, eşitliğin hemen sağındaki `m*g` çarpma işlemindedir. Elde edilen değer sonra `h` değişkeni ile çarpılır.

Örnek 2-7 `c=m*g*h+m*v*v/2` işleminde öncelik, eşitliğin sağındaki ilk terim olan `m*g*h` dedir (buradaki öncelik yine önceki örnekte anlatıldığı gibidir). Daha sonra `m*v*v` çarpma işlemleri, sonra bölme işlemi ve son olarakta toplama işlemi yapılır.

2.4.2 Artırma ++ ve Azaltma – – İşlemleri

Bir işlem için geçen süreyi belirlemek, kaç tekrarın yapıldığını veya geriye sayma işleminin yapılması gerektiği durumlarda (sayaç) artırma veya eksiltme işlemleri için farklı ifadeler kullanılabilir. Çizelge 2-5 da C++ programlama dilinde bu işlemlerin kısaltılması amacıyla verilmiş ifadeler bulunmaktadır. Çizelgede, saydırma (sayaç) işlemlerinde kullanılabilecek bu işlemciler bulunmaktadır. Bu ifadelerde gösterilen sembollerin değişkenin solunda veya sağında olması durumuna göre derleyici tarafından farklı yorumlanır.

Çizelge 2-5 C++ programlama dilinde kullanılan atama sembolleri.

Anlamı	C++ da kullanımı
Sola aktar	=
Artır	++
Eksilt	--
Topladıktan sonra aktar	+=
Eksilttikten sonra aktar	-=
Çarptıktan sonra aktar	*=

Değişkenlerin sakladıkları değerlerin artırılması veya azaltılması işlemleri için +, –, * gibi simgeler kullanılır. Toplama, çıkarma ve çarpma simgelerinin değişkenlerin solunda veya sağında olması işlemlerde önemlidir (Çizelge 2-6).

Çizelge 2-6 C++ programlama dilinde önce işlem sonra artırma veya önce artırma sonra işlem.

a++	Önce a değişkenini kullan, sonra a değişkeninin değerini 1 artır.	a=a+1;
++a	Önce a değişkenindeki değeri 1 artır, sonra a değişkenini kullan.	a+1
a--	Önce a değişkenini kullan, sonra a değişkeninin değerini 1 eksilt.	a=a-1;
--a	Önce a değişkeninin değerini 1 eksilt, sonra a değişkenini kullan.	a-1

Örnek 2-8 a = 4 ve b = 5 değerlerine sahip değişkenler için artırma veya eksiltme işlemleri Çizelge 2-7 verilmektedir. Artırma veya eksiltme sembollerinin değişkenlerin sağında veya solunda olmasına dikkat ediniz.

Çizelge 2-7 C++ programlama dilinde artırma veya eksiltme işlemleri (a=4, b=5).

İşlem	İşleme başlamadan önceki değişkenlerdeki değerler			İşlemden sonra değişkenlerdeki değerler		
	a	b	işlem	a	b	c
c=a++ +b	4	5	c=a+b a=a+1	5	5	9
c=++a +b	5	5	a=a+1 c=a-b	5	5	10
c=--a + b--	3	5	a=a-1 c=a+b b=b-1	3	4	8

Çizelge 2-8 de C/C++ da değişik amaçlar için kullanılan semboller verilmektedir.

Çizelge 2-8 C/C++ da önce değer değiştirme sonra işlem veya önce işlem sonra değer artırma.

İşlem	Anlamı	Örnek	Uzun gösterimi
=	Sağdaki veriyi soldaki değişkene aktar	a=10	a=10
+=	Önce topla sonra soldaki değişkene değeri aktar	a+=10	a=a+10
-=	Önce eksilt sonra soldaki değişkene değeri aktar	a-=10	a:=a-10
*=	Önce çarp sonra soldaki değişkene değeri aktar	a*=10	a=a*10
/=	Önce böl sonra soldaki değişkene değeri aktar	a/=10	a=a/10
%=	Önce böl kalan kısmını al sonra soldaki değişkene aktar	a%=10	a=a%10
>>=	Sağa kaydırarak eşitlik	a>>=10	a=a>>10
<<=	Sola kaydırarak eşitlik	a<<=10	a=a<<10
&=	Ve koşulu ile karşılaştır	a&=10	a=a&10
^=	Veya koşulu ile karşılaştır	a^=10	a=a^10
!=	Veya koşulu ile karşılaştır	a!=10	a=a|10

Çizelgedeki % sembolü modüler bölüm işlemcisidir. Yani bölüm işlemi sonunda kalan değeri verir. Örnek olarak 10%3 = 1 (10 sayısı 3 e bölününce kalan 1 dir), 1%2=1, 9%5=4 , 10%5=0 verilebilir.

Örnek 2-9 Çizelge 2-8 de gösterilen işlemleri yapan bir program aşağıda verilmektedir.

Program 2-3 Çizelge 2-8 deki işlemcilerin kullanımını gösteren C programı.

```
#include <stdio.h>
#include <conio.h>
main () { int a=4, b=5, c;
  printf(" a=%2d ve b=%2d \n", a, b);
  printf("---------------------------------\n");
  c=a+b;
  printf(" Değişkenlerin toplanması          : %d + %d = %d \n", a, b, c);
  a=4; b=5;
  c=a++ +b;
  printf(" Önce topla sonra değişkeni artır  : %d + %d = %d \n", a, b, c);
  a=4; b=5;
  c=a-- +b;
  printf(" Önce eksilt sonra değişkeni azalt : %d + %d = %d \n", a, b, c);
  a=4; b=5;
  c=a*b;
  printf(" çarpma işlemi                     : %d * %d = %d \n", a, b, c);
  a=4; b=5;
  c*=c;
  printf(" Önce çarp                         : %d = \n", c);
  a=4; b=5;
  c/=2;
  printf(" Önce 2 ye böl                     : %d \n", c);
  a=4; b=5;
  c%=3;
  printf(" Bölümden kalanı al                : %d \n", c);
  getch();}
```

Programın çalıştırılması sonucu ekran çıktısı aşağıdaki gibidir:

```
a= 4 ve b= 5
--------------------------------
Değişkenlerin toplanması          : 4 + 5 = 9
Önce topla sonra değişkeni artır  : 5 + 5 = 9
Önce eksilt sonra değişkeni azalt : 3 + 5 = 9
Çarpma işlemi                     : 4 * 5 = 20
Önce çarp                         : 400 =
Ínce 2 ye böl                     : 200
Bölümden kalanı al                : 2
```

2.4.3 Karşılaştırma İşlemleri

Çizelge 2-9 de C++ programlama dilinde sorgulama (iki veya daha fazla koşulun karşılaştırılması) işlemlerinde kullanılan semboller verilmektedir. Sorgulama, koşul veya ilişkisel işlemcilerle, sorgulama koşulunun doğru veya yanlış olmasına bağlı olarak **1** (doğru-true) veya **0** (yanlış, false) sonucu üretilir. Üretilen bu sonuca göre istenilen işleme veya işlemlere devam edilir veya bazı satırlar işleme konulmadan atlanabilir.

Çizelge 2-9 C++ programlama dilinde kullanılan karşılaştırma sembolleri.

Aritmetik gösterimi	Anlamı	C++ da kullanımı	Aritmetik gösterimi	Anlamı	C++ da kullanımı
>	Büyük	>	=	Eşit	==
<	Küçük	<	≠	Eşit değil	!=
≥	Büyük veya eşit	>=	Ve	Ve	&&
≤	Küçük veya eşit	<=	Veya	Veya	||

2.5 ÖNİŞLEMCİ (HEADER) KOMUTLARI

Program satırları içerisinde ihtiyaç duyulan fonksiyonların programa eklenmesi ve kullanılmayacak fonksiyonların programınıza eklenmemesi programcılık açısından daha doğru olacaktır. Rezervasyon veya sipariş programı için trigonometrik fonksiyonlara veya karmaşık sayılara ihtiyaç duyulmaz. Programınız içinde kullanılmak istenilen hazır fonksiyonlar, önişlemciler (header) ile kullanılabilir hale gelir. Burada bahsedilen hazır fonksiyonlar ekrana yazdırma (`printf` veya `cout`), klavyeden veri okuma (`scanf` veya `cin`) komutları olabileceği gibi trigonometrik veya zamanla ilgili fonksiyonları da içerebilir. Programın ilk satırlarına yazılan önişlemci dosyaları programla birlikte derlenir (compiling), bağlanır (linking) ve çalışacak bilgisayar programına eklenir (embedding). Programda kullanılacak içerik dosyası < ve > simgeleri arasında olması durumunda derleyici (compiler) bu dosyaları derleyicinin kurulum aşamasında oluşturulan include isimli bir klasör içinde aranır. Önişlemci dosyası " " simgeleri arasında belirtiliyorsa bu önişlemci dosyaları yazılan program dosyasının bulunduğu klasörde aranır.

Örnek 2-10 Önişlemciler programın ilk satırlarına yazılır ve komut sonlarına ; konulmaz. Aşağıda ekranda gösterme veya klavyeden veri okuması yapabilmek için standart giriş/çıkış (standard input output- stdio.h), standart kütüphaneyi kullanabilmek için stdlib.h, konsoldan (klavye, ekran vs) giriş/çıkış için (consol input/output), matematiksel fonksiyonları kullanabilmek için math.h nin yazımı gösterilmektedir.

```
/*satırın sonuna noktalı virgül konulmaz*/
#include <stdio.h>  veya #include "stdio.h"
```

```
#include <stdlib.h> veya #include "stdlib.h"
#include <conio.h>  veya #include "conio.h"
#include <math.h>   veya #include "math.h"
```

Örnek 2-11 Bir değişkenin sakladığı değer değiştirilmeden program içinde kullanılmak isteniyorsa (bazen değişmez olarak isimlendirilirler) bu tür değişkenler ana programa geçmeden tanımlanabilir.

```
/*define ile başlayan satırın sonuna noktalı virgül konulmaz*/
#define pi 3.14
#define dogru 1
#define yanlis 0
#define topla (a + b) (a + b) /* iki sayının toplamını bulan komut */
#define carp(a * b) (a * b) /* iki sayının çarpımının tanımlanması*/
#define radyan x*(pi/180) /* radyanın tanımlanması */
```

Çizelge 2-10 de `math.h` önişlemci dosyasının içinde bulunan hazır fonksiyonları verilmektedir.

Çizelge 2-10 C++ da kullanılabilecek bazı matematiksel fonksiyonlar (`<math.h>`)

Fonksiyon	İşlem	Fonksiyon	İşlem	Fonksiyon	İşlem
`abs(x)`	mutlak değer	`ceil(x)`	x e en yakın tamsayı	`log(x)`	doğal logaritma
`pow(x,y)`	x in y inci kuvveti (x^y)	`floor(x)`	x ten en uzak tamsayı	`log10(x)`	10 tabanına göre logaritma
`sqrt(x)`	x in karekökü	`exp(x)`	e^x	`sin(x)`	sinüs(x) radyan
`cos(x)`	kosinüs(x) radyan	`tan(x)`	Tanjant(x) radyan	`asin(x)`	arcsinüs(x) radyan
`acos(x)`	Arkkosinüs(x) radyan	`atan(x)`	Arctanjant(x) rad.	`sinh(x)`	sinüs hiperbolik rad.
`cosh(x)`	Cosinüs hiperbolik	`tanh(x)`	Tanjant hiperbolik rad.		

Örnek 2-12 Aşağıdaki programda bazı hazır fonksiyonların kullanımı gösterilmektedir. `using namespace std;` komutu önişlemci dosyalarındaki h lerin yazılmamasını sağlamaktadır. Satır sonlarındaki endl; komutu ekrandaki imleçin 1 satır aşağı inmesini sağlar (yeni satır).

Program 2-4 Hazır fonksiyonların kullanımı.

```
#include <cmath>
#include <iostream>
```

```
using namespace std; // .h lerin yazılmamasını sağlar
int main() {
double x = 2.718281828459045;
cout << x << endl;
cout << "ceil(x)     = " << ceil(x) << endl;
cout << "floor(x)    = " << floor(x) << endl;
cout << "sqrt(x)     = " << sqrt(x) << endl;
cout << "pow(x, 0.5) = " << pow(x, 0.5) << endl;
cout << "log(x)      = " << log(x) << endl;
cout << "log10(x)    = " << log10(x) << endl;
cout << "exp(x)      = " << exp(x) << endl;
cout << "sin(x)      = " << sin(x) << endl;
system("PAUSE"); // programın ekranda kalmasını sağlar }
```

Programın çalıştırılması sonucu ekran çıktısı aşağıdaki gibidir:

```
2.71828
ceil(x)     = 3
floor(x)    = 2
sqrt(x)     = 1.64872
pow(x, 0.5) = 1.64872
log(x)      = 1
log10(x)    = 0.434294
exp(x)      = 15.1543
sin(x)      = 0.410781
Press any key to continue . . .
```

2.5.1 C++ Açıklama Satırları

Yazdığınız programın daha anlaşılır olabilmesi için bazı yerlerinde açıklamaların yapılması çok yararlı olacaktır. Bunun için C++ da // simgeleri ile, satırın herhangi bir yerinden açıklama yapılmaya başlanabilir ve satırın bittiği yere kadar açıklamalara devam edilir. ANSI-C de tanımlanan (eski) açıklama satırı aşağıdaki gibi kullanılmaktadır:

`/* ... */` veya `// ....//`

2.5.2 Blok

C/C++ da komutlardan veya satırlardan oluşan bir grubu tanımlamak için **blok** kelimesi kullanılır. Blok { (sol küme parantezi) ve } (sağ küme parantezi) simgeleri arasındadır. Örnek olarak

```
{ durum1;
  durum2;
  durum3;
  ... }
```