

ALGORİTMA

Belli bir problemi çözmek veya belli bir amaca ulaşmak için geliştirilmiş sisteme *algoritma* denir. Bilgisayar yardımıyla çözülecek bir problemin, yazılacak bir programın aşağıdaki aşamalardan geçmesi gerekir:

1. Problemin tanımı
2. Çözüm yolunun tespiti
3. Algoritmanın hazırlanması
4. Akış diyagramının çizilmesi
5. Programın hazırlanması
6. Hazırlanan programın test edilmesi
7. Uygulama

Örnek : İki sayının toplama işlemini yapan programın algoritmasını yazınız.

1. BAŞLA
2. A değerinin girilmesi
3. B değerinin girilmesi
4. $TOPLAM=A + B$ işlemini yap
5. $TOPLAM$ değerini ekrana yaz
6. SON

Örnek : Klavyeden girilecek iki sayıdan büyük olanından küçük olanını çıkarıp sonucu ekrana yazacak program için bir algoritma geliştiriniz.

- A. BAŞLA
- B. A değerinin girilmesi
- C. B değerinin girilmesi
- D. Eğer A büyüktür B $SONUC=A-B$
Değilse $SONUC=B-A$
- E. $SONUC$ değerini ekrana yaz
- F. SON

Örnek: Bir öğrenciye ait iki farklı not bilgisini alarak, not ortalamasını hesaplayan algoritma.

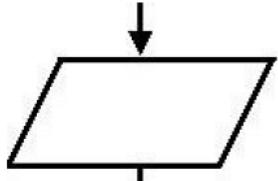
Öncelikle değişkenler belirlenir.

Not1 = N1;

Not2 = N2;

ORTALAMA = ORT;

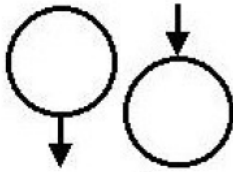
1. Başla;
2. Notun girilmesi (N1);
3. Notun girilmesi (N2)
4. Not ortalamasının hesaplanması ($ORT=(N1+N2)/2$)
5. Ortalama sonucunun görüntülenmesi(ORT)
6. Son.



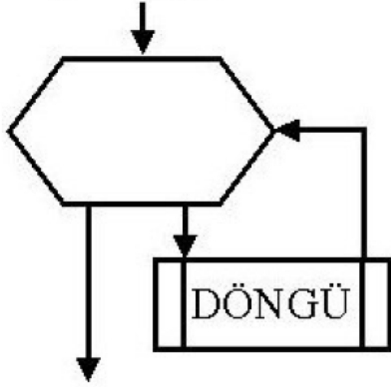
Giriş Çıkış komutunun kullanılacağı yeri belirler ve kutu içerisine hangi değişkeni ve OKU mu YAZ mı yapılacağını belirtmeniz gerekir



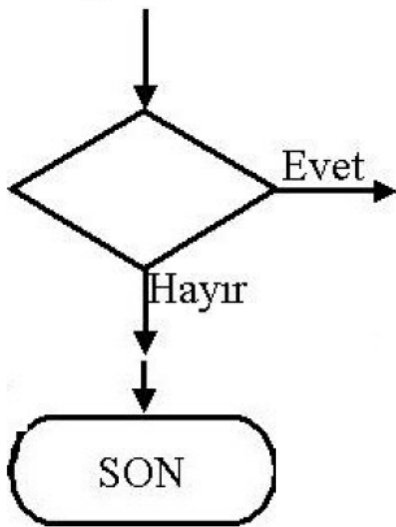
Bilginin Yazıcıya yazılacağı konumu gösteren şekildir.



Bir algoritmanın birden fazla alana yayılması durumunda bağlantı noktalarını gösteren şekildir. Tek girişli veya tek çıkışlı olarak kullanılırlar.



Bir işlemin belli bir sayıda veya belli bir koşul doğru olduğu sürece tekrar edilmesini sağlayan döngü komutunu gösteren şekildir. Bu döngüde altıgen içerisine ya koşul ya da döngünün başlangıç, adım ve sonlanma değerlerini belirtebilirsiniz. DÖNGÜ olarak belirlenen blokta da tekrar edilmek istenen komutlar yer almaktadır.



Bir algoritmada bir kararın verilmesini ve bu karara göre iki seçenekten birinin uygulanmasını sağlayan şekildir. burada eşkenar dörtgen içerisine kontrol edilecek mantıksal koşul yazılır. Program akışı sırasında koşulun doğru olması durumunda "Evet" yazılan kısma Yanlış olması durumunda "Hayır" yazılan kısma sapılır. Tek girişli ve çift çıkışlı bir şekildir.

Programın bittiği yer ya da yerleri gösteren bir şekildir.

ALGORİTMA VE AKIŞ ŞEMASINA GENEL ÖRNEKLER

ÖRNEK: İki sayının çarpımının bulunmasıyla ilgili algoritma şöyledir.

Değişkenler:

A: Birinci sayıyı

B: İkinci sayıyı

C: İki sayının çarpımını ($A*B$) göstereceğiz

Algoritma:

Adım 1- Başla

Adım 2- A'yı oku

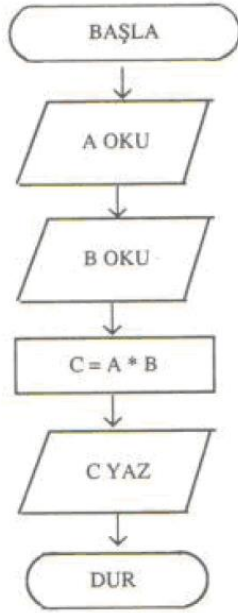
Adım 3- B'yi oku

Adım 4- $C=A*B$ yi hesapla

Adım 5- C'yi yaz

Adım 6- Dur

Akış Şeması :



ÖRNEK: İki sayının farkını ve bölümünü bulup yazıcı ile yazan algoritma ve akış şeması şöyledir.

Değişkenler:

A: Birinci sayıyı tutmak için tanımlayacağımız

B: İkinci sayıyı

D: İki sayının farkını ($A-B$)

E: İki sayının bölümünü (A/B) göstereceğiz

Algoritma:

Adım 1- Başla

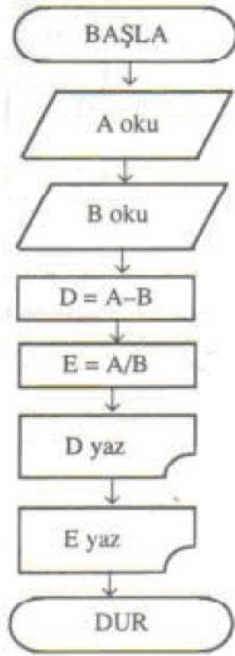
Adım 2- A'yı oku

Adım 3- B'yi oku

Adım 4- $D=A-B$ yi hesapla

- Adım 5- $E=A/B$ yi hesapla
Adım 6- D'yi yaz
Adım 7- E'yi yaz
Adım 8- Dur

Akış şeması:



ÖRNEK: İki sayının toplamlarının karesini ve küpünü hesaplayıp yazıcı ile yazan akış şeması şöyledir.

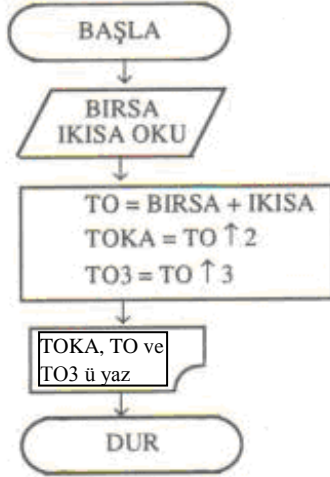
Değişkenler:

- BIRSA: Birinci sayıyı
İKISA: İkinci sayıyı
TO: Toplamı
TOKA: Toplamın karesini
TO3: Toplamın küpünü gösterebilirsin

Algoritma:

- Adım 1- Başla
Adım 2- BIRSA ve İKISA'yı oku
Adım 3- $TO=BIRSA+İKISA$
 $TOKA=TO^2$
 $TO3=TO^3$
Adım 4- TOKA, TO'yu ve TO3'ü yaz
Adım 5- Dur

Akış şeması:



ÖRNEK: A ve B gibi iki sayıdan büyüğünü printerle(yazıcı) yazdıran algoritma ve akış şeması şöyledir.

Değişkenler:

A: Birinci sayı

B: İkinci sayı

Algoritma:

Adım 1-Başla

Adım 2-A,B'yi oku

Adım 3-A=B ise Adım 7'ye git

Adım 4-A>B ise Adım 6'ya git

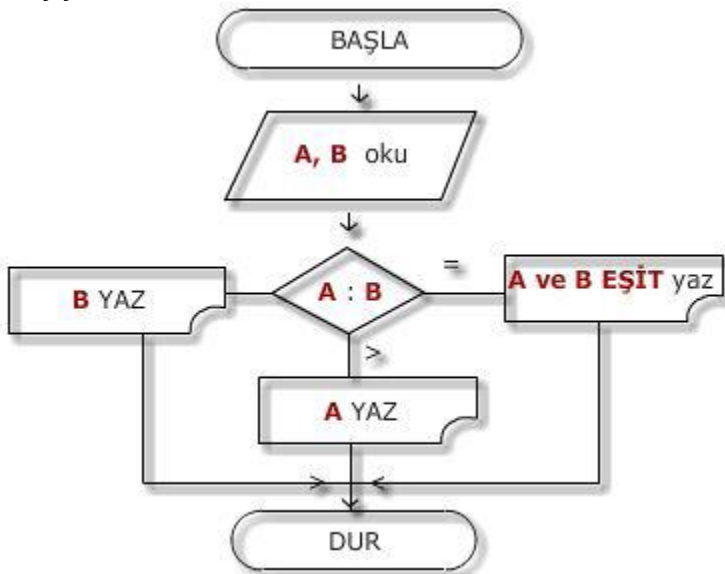
Adım 5-B'yi yaz Adım 8'e git

Adım 6-A'yı yaz Adım 8'e git

Adım 7-"A ve B eşit" mesajını yaz, Adım 8 e git

Adım 8-Dur

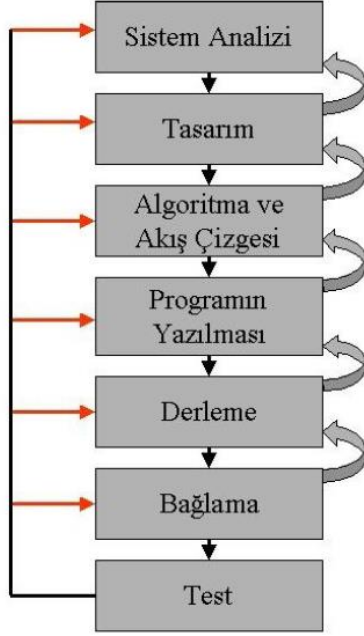
Akış şeması:



Örnek sorular

- 1- Girilen iki sayı üzerinde dört işlem yapan programın algoritmasını yazınız. Akış şemasını çiziniz.
- 2- Bir öğrencinin vize ve final notunu alarak geçme notunu hesaplayınız. Akış şemasını çiziniz.

Yazılım Geliştirme



Yazılım (software): Programlama ve programlama ile ilgili konuların geneline verilen isimdir. Yazılım denince ilk olarak akla programlama dilleri, bu diller kullanılarak yazılmış kaynak programlar ve çeşitli amaçlar için oluşturulmuş dosyalar gelmektedir.

Uygulama yazılımı dillerine örnek olarak, Basic, Pascal, Delphi, Visual Basic, C, C++, C#, Java v.b.

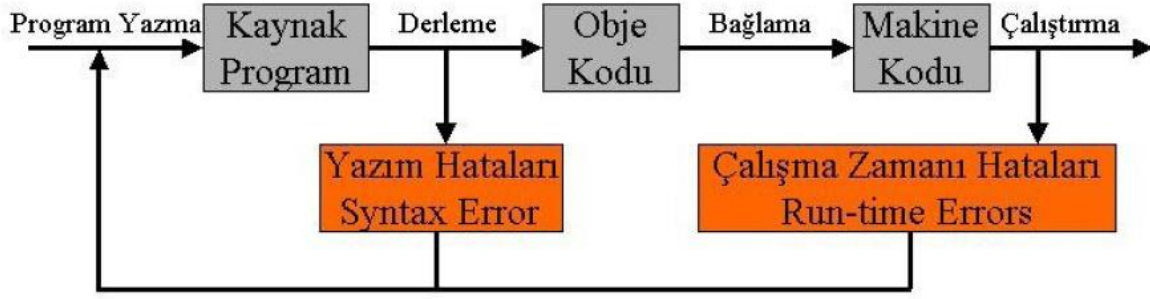
PROGRAM NEDİR? PROGRAMLAMA DİLİ NEDİR?

Program, yapılmasını istediğiniz işlemleri bilgisayara bildirdiğiniz adım adım komutlar kümesidir. Bilgisayara istediğiniz işlemleri bildirme yolunu sağlayan kurallar kümesi bir programlama dilini oluşturmaktadır. Pek çok programlama dili vardır.

Derleyici Nedir?

Bir programlama dili ile bilgisayara aktarılan programın bilgisayarın anlayabileceği Makine Diline çevirmeyi sağlayan ve yazılan programda söz dizim hatalarının olup olmadığını bulan yazılımlardır. Her Programlama dili için bir derleyici olması gerekmektedir.

Örnek olarak; DevC++, Visual Studio, Turbo C v.b.



C PROGRAMLAMA DİLİ

C Programlama Dili genel amaçlı orta seviyeli ve yapısal bir programlama dilidir. 1972 yılında Dennis Ritchie tarafından Bell Telefon Laboratuvarında 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.

Günümüzde nesneye yönelik programlama dilleri (C++, Java) ve script dilleri (JavaScript, JavaApplet, PHP) gibi programlama dilleri C Programlama Dili'nden esinlenmiştir.

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 adlandırılan alt programlardan oluşmuştur.
- C++, Java, JavaScript, JavaApplet, PHP, C#, ... gibi diller C dilinden esinlenmiştir.

Örnek : Derlendikten sonra ekrana 'Merhaba Dünya!' yazar

```

01:  /* ilk.c: ilk C programi */
02:  // üstteki bölü ve yıldız işaretleri uzun cümleler için , // işareti tek satır açıklamalar için
03:
04:  #include <stdio.h>
05:
06:  int main()
07:  {
    printf("Merhaba Dünya!\n");
  }
  
```

#include <stdio.h>

Kütüphane dosyasıdır. Bunun *haricinde* (*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*) de vardır.

main()

6. 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()

8. 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 kullanımını gösterilmiştir.

Örnek kullanım şekli

```
printf("Element: Aluminyum");  
printf("Atom numarası = %d",13);  
printf("Yoğunluk = %f g/cm3",2.7);  
printf("Erime noktası = %f derece",660.32);
```

Ekranda yazılacak ifade

```
Element: Aluminyum  
Atom numarası = 13  
Yoğunluk = 2.7 g/cm3  
Erime noktası = 660.32 derece
```

Kaynak Kodunun Derlenmesi

İşletim Sistemi	Derleyici	Derleme	Çalıştırma
MS-DOS / Windows	Microsoft C	cl ilk.c	ilk.exe
	Borland Turbo C <u>Web</u>	tcc ilk.c	ilk.exe
Windows 7	DEV C++	Main.c	Main.exe

C Kodlarının Temel Özellikleri

Bir C programı aşağıda verilen özellikleri mutlaka taşımalıdır.

- Yazılımda kullanılacak olan her fonksiyon için ilgili başlık dosyası programın başına ilave edilmelidir.
- 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.

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 olmamalıdır. Eğer bu isimler bir kaç kelimedenden 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.

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 nitelendiriciler vardır ki bunlar yukarıdaki temel tiplerin önüne gelerek onların türevlerini oluşturur. Bunlar:

```
short
long
unsigned
```

Örnek :

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

```
main()
{
    printf( "char      : %d bayt\n", sizeof(char));
    printf( "short     : %d bayt\n", sizeof(short));
    printf( "int       : %d bayt\n", sizeof(int));
    printf( "long      : %d bayt\n", sizeof(long));
    printf( "unsigned char : %d bayt\n", sizeof(unsigned char));
    printf( "unsigned short : %d bayt\n", 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));
}
```

Örneğin Ekran Çıktısı :

```
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
```

NOT : BU TABLOYU EZBERLEMENİZ GEREKMEZ.

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

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 değer nasıl bir veri olduğunu gösteren bir veri tipi vardır

Sabitler

Sabit bildirim, başlangıç değeri verilen değişken bildirim 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;
```

Ya da sabitler kütüphanelerin altında tanımlanabilir.

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

Değişkenlerin ve sabitlerin ekrana yazdırılması

```
01: /* 02prg02.c : Değişkenler ve sabitlerin ekrana yazdırılması
02: */
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;
}
```

Değişken Bildirim Yerleri ve Türleri

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

İnt a; // genel bildirimdir her yerde kullanılabilir bu değişken

```
int main()
{
```

İnt b; // yerel bildirimdir. Bu değişken sadece main içinde kullanılabilir.

```
}
```

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.

Örneğin:

```
int x=9;
```

```
float a,b,c;
```

```
double d;
```

```
...
```

```
a = x/4;
```

```
b = x/4.0;
```

```
c = (float) x/4; // bu satırdaki kodda x/4 işleminin sonucu float a
```

```
dönüştürülmüştür.Çünkü sonucun verildiği değişkenin yani c nin tipi floattır.
```

Aritmetik Operatörler

Değişken veya sabitler üzerinde temel aritmetik işlemleri gerçekleyen operatörlerdir. Bunlar Tablo 3.1'de listelenmiştir.

Aritmetik Operatörler

Operatör	Açıklama	Örnek	Anlamı
+	toplama	$x + y$	x ve y nin toplamı
-	çıkarma	$x - y$	x ve y nin farkı
*	çarpma	$x * y$	x ve y nin çarpımı
/	bölme	x / y	x ve y nin oranı
%	artık bölme	$x \% y$	x / y den kalan sayı

Atama Operatörleri

Operatör	Açıklama	Örnek	Anlamı
=	atama	$x = 7;$	$x = 7;$
+=	ekleyerek atama	$x += 3$	$x = x + 3$
-=	eksilterek atama	$x -= 5$	$x = x - 5$
*=	çarparak atama	$x *= 4$	$x = x * 4$
/=	bölerek atama	$x /= 2$	$x = x / 2$
%=	bölüp, kalanını atama	$x \% = 9$	$x = x \% 9$
++	bir arttırma	$x++$ veya $++x$	$x = x + 1$
--	bir azaltma	$x--$ veya $--x$	$x = x - 1$

Kontrol karakterleri

Karakter	Anlamı
\a	Ses üretir (alert)
\b	imleci bir sola kaydır (backspace)
\f	Sayfa atla. Bir sonraki sayfanın başına geç (formfeed)
\n	Bir alt satıra geç (newline)
\r	Satır başı yap (carriage return)
\t	Yatay TAB (horizontal TAB)
\v	Dikey TAB (vertical TAB)
\"	Çift tırnak karakterini ekrana yaz
\'	Tek tırnak karakterini ekrana yaz
\\	\ karakterini ekrana yaz
%%	% karakterini ekrana yaz

Tip karakterleri

Tip Karakteri	Anlamı	Yazdırılacak veri tipi
%c	tek bir karakter	char
%s	karakter dizisi (string)	char*
%d	işaretli ondalık tamsayı	int, short
%ld	uzun işaretli ondalık tamsayı	long
%u	işaretsiz ondalık tamsayı	unsigned int, unsigned short
%lu	işaretsiz uzun tamsayı	unsigned long
%f	Gerçel sayı	float
%lf	Çift duyarlı gerçel sayı	double

printf() Fonksiyonu

Standart C kütüphanesinde bulunan printf() fonksiyonu, değişkenlerin tuttuğu değerleri, onların adreslerini veya bir mesajı ekrana belli bir düzenle (format) standart çıkışa (stdout), yani ekrana, yazdırmak için kullanılan fonksiyondur. Daha önce yazılan örnek programlarda printf() fonksiyonundan yararlanmıştık. Şimdi bu fonksiyonun nasıl kullanıldığına bakalım.

Genel yazım biçimi:

```
int printf(const char *format, ...);
```

Basit olarak ekrana Hata oluştu!.. şeklinde bir mesaj yazdırma işlemi:

```
printf("Hata Oluştı!..");
```

şeklindedir. Çoğu zaman ekrana, programda kullanılan bir değişkenin değeri yazdırılmak istenebilir. Örneğin ekrana bir tamsayı değişkeninin içeriğini basırmak için, printf()

....

```
int x = 12;
printf("x in deęeri %d dir", x);
```

....

gibi kullanılır. Bu program parçasının ekran çıktısı şöyle olacaktır:
x in deęeri 12 dir

Bu örnekte printf fonksiyonuna iki parametre aktarılmıştır. Birincisi ekranda gösterilecek ve çift tırnaklar arasına yazılan ifadeler, ikincisi ise ekranda sayısal deęeri gösterilmek istenen deęişken (x).

**format* üç kısımdan oluşmaktadır:

- I. **Düz metin (literal string)**: yazdırılmak istenen ileti.
Örneęin: printf("Ben gelmedim kavga için..."); gibi.
- II. **Konrol karakterleri (escape squence)**: deęişkenlerin ve sabitlerin nasıl yazılacağını belirtmek veya imlecin alt satıra geçirilmesi gibi bazı işlemlerin gerçekleştirilmesi için kullanılır. Bu karakterler Tablo 4.1'de listelenmiştir.
Örneęin: printf("\tDostun evi gönlüdür...\n"); gibi.

scanf() Fonksiyonu

Birçok programda ekrana verilerin bastırılmasının yanısıra klavyeden veri okunması gerekebilir. scanf() fonksiyonu klavyeden veri okumak için kullanılan fonksiyondur. printf() gibi scanf() fonksiyonunda Tablo 4.1 ve Tablo 4.2'de verilen karakterleri kullanır. Örneęin klavyeden bir x tamsayısı okumak için:

```
scanf("%d",&x);
```

satırını yazmak yeterli olacaktır. Burada & işareti *adres operatörü* olarak adlandırılır ve [Böüm 11](#)'de ayrıntılı olarak açıklanacaktır. Klavyeden iki farklı sayı okunmak istendięinde scanf() fonksiyonu şöyle kullanılabilir:

```
scanf("%d %f",&x,&y);
```

veriler klavyeden

```
16 1.56
```

yada

```
16 1.56
```

veya

```
16
```

```
1.56
```

şeklinde girilebilir.

puts () Fonksiyonu

Ekrana yazdırılacak ifade bir karakter topluluęu ise, printf() 'e alternatif olarak puts () fonksiyonu kullanılabilir. Ancak puts (), ekrana bu karakter topluluęu yazdıktan sonra, imleci alt satıra geçirir. Buna göre:

```
printf("Sevgi varlıęın mayasıdır.\n");
```

ile

```
puts("Sevgi varlığın mayasıdır.");  
kullanımları eşdeğerdir.
```

gets () Fonksiyonu

Klavyeden bir karakter topluluğu okumak için kullanılır. Okuma işlemi yeni satır karakteriyle(\n) karşılaşıncaya kadar sürer. puts () - gets () arasındaki ilişki, printf () - scanf () arasındaki gibidir. Yani,

```
scanf("%s",&str);  
ile  
gets(str);
```

getchar () Fonksiyonu

Bu fonksiyon ile standart girişten bir karakter okunur. Programı istenen bir yerde durdurup, bir karakter girinceye kadar bekletir. Örneğin:

```
...  
for(i=0; i<10; i++)  
{  
    getchar();  
    printf("%d\n",i);  
}  
...
```

Yukarıdaki program parçası 0-9 arası sayıları sırasıyla ekranda göstermek için kullanılır. Fakat her rakamı yazdırılmadan önce klavyeden herhangi bir karakter girip [Enter] tuşuna basılması beklenir. Bu bekleme getchar () fonksiyonu ile gerçekleştirilir.

Formatlı Çıktı

Bundan önceki programlardaki değişkenler serbest biçimde (free format), yani derleyicinin belirlediği biçimde ekrana yazdırılmıştı. Bazen giriş ve çıkışın biçimi kullanıcı tarafından belirlenmesi gerekebilir. Bu işlem:

Tamsayılarda %d yerine %wd
Gerçel sayılarda %f yerine %w.kf
Stringlerde %s yerine %ws

biçimindeki kullanım ile sağlanır. Burada w yazılacak olan sayının alan genişliği olarak adlandırılır. Gerçel bir değişken ekrana yazılacaksa, değişkenin virgülden sonra kaç basamağının yazdırılacağı k sayısı ile belirlenir. Ancak $w > k + 2$ olmalıdır.

```
int i=583,j=1453;  
  
printf("%d %d\n",i,j);    /* serbest biçim */
```

```
printf("%5d %8d\n",i,j); /* formatlı */
program parçasının ekran çıktısı şöyledir:
```

ÇIKTI

```
583 1453
583    1453
```

Gerçel sayılarda iş biraz daha karışık. Örneğin:

```
int x=123.456;
```

```
printf("%f\n",x); /* serbest biçim */
printf("%8.2f\n",x); /* formatlı */
```

program parçası çalıştırıldığında aşağıdaki sonuç gözlenir:

ÇIKTI

```
123.456001
123.46
```

Matematiksel Fonksiyonlar (math.h)

Matematiksel fonksiyonların hemen hemen hepsi double veri tipindedir. Bu fonksiyonlardan biri program içinde kullanılacaksa math.h başlık dosyası program içine eklenmelidir.

Fonksiyon Bildirimi	Açıklama	Örnek	Sonuç
int abs(int x);	x tamsayısının mutlak değerini hesaplar	abs(-4)	4
double fabs(double x);	x gerçel sayısının mutlak değerini hesaplar	fabs(-4.0)	4.000000
int floor(double x);	x'e (x'den büyük olmayan) en yakın tamsayıyı gönderir	floor(-2.7)	-3
int ceil(double x);	x'e (x'den küçük olmayan) en yakın tamsayıyı gönderir	ceil(-2.7)	-2
double sqrt(double x);	pozitif x sayısının karekökünü hesaplar	sqrt(4.0)	2.000000
double pow(double x, double y);	x ^y değerini hesaplar	pow(2., 3.)	8.000000
double log(double x);	pozitif x sayısının doğal logaritmasını hesaplar, ln(x)	log(4.0)	1.386294
double log10(double x);	pozitif x sayısının 10 tabanındaki logaritmasını hesaplar	log10(4.0)	0.602060
double sin(double x);	radyan cinsinden girilen x sayısının sinüs değerini hesaplar	sin(3.14)	0.001593
double cos(double x);	radyan cinsinden girilen x sayısının kosinüs değerini hesaplar	cos(3.14)	-0.999999
double tan(double x);	radyan cinsinden girilen x sayısının tanjant değerini hesaplar	tan(3.14)	-0.001593
double asin(double x);	sinüs değeri x olan açığı gönderir. Açık -pi/2 ile pi/2 arasındadır	asin(0.5)	0.523599
double acos(double x);	cosinüs değeri x olan açığı gönderir. Açık -pi/2 ile pi/2 arasındadır	acos(0.5)	1.047198
double atan(double x);	tanjant değeri x olan açığı gönderir. Açık -pi/2 ile pi/2 arasındadır	atan(0.5)	0.463648
M_PI	Değeri 3.14159265358979323846 olan ön tanımlı sembolik sabit		
M_E	Değeri 2.7182818284590452354 olan ön tanımlı sembolik sabit		

Standart Kütüphane Fonksiyonları (stdlib.h)

Standart kütüphanede, programı sonlandıran, dinamik bellek yönetiminde kullanılan veya rastgele sayı üretme vb. işlevleri yerine getiren bir çok fonksiyon mevcuttur. Bu kısımda, bunlardan bir kaçını Tablo 5.2'de listelenmiştir.

stdlib.h kütüphanesinde tanımlı bazı fonksiyonlar

Fonksiyon Bildirimi	Açıklama	Örnek	Sonuç
<code>int atoi(const char *s);</code>	Bir karakter topluluğunu tamsayıya çevirir	<code>atoi("-12345")</code>	-12345
<code>long atol(const char *s);</code>	Bir karakter topluluğunu uzun tamsayıya çevirir	<code>atol("1234567890")</code>	1234567890
<code>double atof(const char *s);</code>	Bir karakter topluluğunu gerçel sayıya çevirir	<code>atof("-123.546")</code>	-123.456
<code>void exit(int durum);</code>	Programı sonlandırarak kontrolü işletim sistemine geri verir.	<code>exit(1)</code>	-
<code>int rand(void);</code>	0 ile RAND_MAX arasında rastgele sayı üretir. RAND_MAX, stdlib.h içinde tanımlanmış bir sembolik sabittir	<code>rand()</code>	50485132
<code>max(a,b)</code>	stdlib.h'de tanımlanmış iki sayıdan en büyüğünü bulan makro fonksiyon	<code>max(5, 9)</code>	9
<code>min(a,b)</code>	stdlib.h'de tanımlanmış iki sayıdan en küçüğünü bulan makro fonksiyon	<code>min(5, 9)</code>	5

Karakter Üzerinde İşlem Yapan Fonksiyonlar (ctype.h)

ctype.h Kütüphanesinde tanımlı fonksiyonlar

Fonksiyon Bildirimi	Açıklama	Örnek	Sonuç
<code>isalpha(c)</code>	c bir harf ise 0 dan farklı, değilse 0 gönderir	<code>isalpha('a')</code>	8
<code>isalnum(c)</code>	c A-Z, a-z veya 0-9 arasında ise 0 dan farklı, değilse 0 gönderir	<code>isalnum('a')</code>	1
<code>isascii(c)</code>	c bir ASCII karakter ise 0 dan farklı, değilse 0 gönderir	<code>isascii('a')</code>	1
<code>isdigit(c)</code>	c bir rakam ise 0 dan farklı, değilse 0 gönderir	<code>isdigit('4')</code>	2
<code>islower(c)</code>	c a-z arasında ise 0 dan farklı, değilse 0 gönderir	<code>islower('P')</code>	0
<code>isupper(c)</code>	c A-Z arasında ise 0 dan farklı, değilse 0 gönderir	<code>islower('P')</code>	4
<code>toascii(c)</code>	c sayısı ile verilen ASCII koda sahip karakteri elde eden makro	<code>toascii(65)</code>	A
<code>tolower(c)</code>	c karakterini küçük harfe çevirir	<code>tolower('D')</code>	d
<code>toupper(c)</code>	c karakterini büyük harfe çevirir	<code>toupper('b')</code>	B

Karşılaştırma Operatörleri ve Mantıksal Operatörler

Karşılaştırma Operatörleri, sayısal değerleri veya karakterleri mukayese etmek için kullanılır.

Karşılaştırma Operatörleri

Operatör	Açıklama	Örnek	Anlamı
>	büyüktür	$x > y$	x, y den büyük mü?
<	küçüktür	$x < y$	x, y den küçük mü?
==	eşittir	$x == y$	x, y ye eşit mi?
>=	büyük-eşittir	$x >= y$	x, y den büyük yada eşit mi?
<=	küçük-eşittir	$x <= y$	x, y den küçük yada eşit mi?
!=	eşit değil	$x != y$	x, y den farklı mı?

Birden çok karşılaştırma işlemi, Tablo 6.2'deki Mantıksal Operatörler'le birleştirilebilir.

Mantıksal Operatörler

Operatör	Açıklama	Örnek	Anlamı
&&	mantıksal VE	$x > 2 \ \&\& \ x < y$	x, 2 den büyük VE y den küçük mü?
	mantıksal VEYA	$x > 2 \ \ x < y$	x, 2 den büyük VEYA y den küçük mü?
!	mantıksal DEĞİL	$!(x > 2)$	x, 2 den büyük değilse

C dilinde, bir mantıksal işlemin sonucu tamsayı 0 (sıfır) veya başka bir değer olur. 0 *olumsuz* 0'dan farklı değerler *olumlu* olarak yorumlanır. Buna göre, aşağıdaki program parçasının

```

...
int x = 1, y = 2, s, u, z;

    s = 2 > 1;
    u = x > 3;
    z = x <= y && y > 0;

    printf("%d\t%d\t%d", s, u, z);
...

```

çıktısı:

```
1      0      1
```

şeklinde olur. Bunun nedeni:

- 2 her zaman 1 den büyük olduğu için s değişkenine 1,
- $x = 1 < 3$ olduğu için x değişkenine 0,
- $z = x <= y \ \&\& \ y > 0$; eşitliğin sağtarafının sonucu olumlu olduğu için z değişkenine 1 atanır.

if, if-else Yapısı

Bu deyimler, koşullu işlem yapan deyimlerdir. if ve else tek bir karşılaştırma deyimini olup else kullanımı isteğe bağlıdır. Eğer bu koşul olumlu ise if den sonraki bölüm yürütülür ve else den sonraki bölüm atlanır. Koşul olumsuz ise if den sonraki küme atlanır ve eğer varsa, else den sonraki kümedeki işlemler gerçekleştirilir.

if deyiminin yapının genel biçimi şöyledir:

```

if (koşul)
{
    ...
    deyimler; (küme)
    ...
}

```

```
}
```

`if` deyimi kullanılırken kümenin başlangıcı ve bitişini gösteren, küme parantezleri kullanılmasında kullanıcıya bir esneklik sunulmuştur. Eğer `if` deyiminden sonra icra edilecek deyimler tek satırdan oluşuyorsa, bu işaretlerin kullanılması zorunlu değildir. Yani, `if` deyimden sonra `{` ve `}` işaretleri kullanılmamışsa, bu deyimi takip eden sadece ilk satır işleme konur. Bu durum, `else if`, `else` deyimlerinde ve daha sonra işlenecek `for` ve `while` gibi döngü deyimlerinde de geçerlidir.

Buna göre aşağıdaki kullanım

```
if(x == y){  
    puts("x ve y esit");  
}
```

ile

```
if(x == y)  
    puts("x ve y esit");
```

eşdeğerdir.

```
if(koşul_1)  
{  
    ...  
    Koşula bağlı olarak yapılacak işlemler;  
    ...  
}  
else if(koşul_2)  
{  
    ...  
    Koşula bağlı olarak yapılacak işlemler;  
    ...  
}  
.  
.  
.  
else if(koşul_n-1)  
{  
    ...  
    Koşula bağlı olarak yapılacak işlemler;  
    ...  
}  
else  
{  
    ...  
    Koşula bağlı olarak yapılacak işlemler;  
    ...  
}
```

switch - case Yapısı

Bu deyim bir *değişkenin* içeriğine bakarak, programın akışını bir çok seçenektan birine yönlendirir. `case` (durum) deyiminden sonra değişkenin durumu belirlenir ve takip eden gelen satırlar (deyimler) işleme konur. Bütün durumların aksi söz konu olduğunda

gerçekleştirilmesi istenen deyimler default deyiminden sonraki kısımda bildirilir. Genel yazım biçimi:

```
switch (değişken)
{
    case sabit1:
        ...
        deyimler;
        ...
    case sabit2:
        ...
        deyimler;
        ...
    case sabitn:
        ...
        deyimler;
        ...
    default:
        ...
        hata deyimleri veya varsayılan deyimler;
        ...
}
```

Switch case ile IF karşılaştırması

<pre>switch(secim) { case 1: sonuc = x + y; printf("Toplam = %f\n", sonuc); break; case 2: sonuc = x-y; printf("Fark = %f\n", sonuc); break; case 3: sonuc = x * y; printf("Carpim = %f\n", sonuc); break; case 4: sonuc = x/y; printf("Oran = %f\n", sonuc); break; default: puts("Yanlis secim !\a"); }</pre>	<pre>if(secim == 1){ sonuc = x + y; printf("Toplam = %f\n", sonuc); } else if(secim == 2){ sonuc = x-y; printf("Fark = %f\n", sonuc); } else if(secim == 3){ sonuc = x * y; printf("Carpim = %f\n", sonuc); } else if(secim == 4){ sonuc = x/y; printf("Oran = %f\n", sonuc); } else{ puts("Yanlis secim !\a"); }</pre>
---	--

? Karşılaştırma Operatörü

Bu operatör, `if-else` karşılaştırma deyiminin yaptığı işi sınırlı olarak yapan bir operatördür. Genel yazım biçimi:

```
(koşul) ? deyim1 : deyim2;
```

İlk önce koşul sınanır. Eğer koşul olumluysa `deyim1` aksi takdirde `deyim2` değerlendirilir. `deyim1` ve `deyim2` de atama işlemi yapılamaz. Ancak `koşul` deyiminde atama işlemi yapılabilir. `deyim1` ve `deyim2` yerine fonksiyon da kullanılabilir. Aşağıda bu deyim kullanımına ait örnekler verilmiştir.

```
x = ( a > b ) ? a : b;
```

Yukarıdaki ifadede koşul `a`'nın `b`'den büyük olmasıdır. Eğer olumluysa `x` adlı değişkene `a`, değilse `b` değeri atanır. Bu şekilde kullanım `if-else` yapısı ile kurulmak istenirse:

```
if( a > b ) x = a;  
else      x = b;
```

while Döngüsü

Tekrarlama deyimidir. Bir küme ya da deyim `while` kullanılarak bir çok kez yinelenebilir. Yinelenmesi için koşul sınaması döngüye girilmeden yapılır. Koşul olumlu olduğu sürece çevrim yinelenir. Bu deyim kullanımını Program 7.1 de gösterilmiştir. Genel yazım biçimi:

```
while(koşul)  
{  
    ...  
    Koşula bağlı olarak yapılacak işlemler;  
    ...  
}
```

do ... while Döngüsü

Bu deyim `while` deyiminden farkı, koşulun döngü sonunda sınanmasıdır. Yani koşul sınanmadan döngüye girilir ve döngü kümesi en az bir kez yürütülür. Koşul olumsuz ise döngüden sonraki satıra geçilir. Bu deyim kullanımını Program 7.2 de gösterilmiştir. Genel yazım biçimi:

```
do{  
    ...  
    Koşula bağlı olarak yapılacak işlemler;  
    ...  
}while(koşul);
```

for Döngüsü

Bu deyim, diğer döngü deyimleri gibi bir kümeyi bir çok kez tekrarlamak için kullanılır. Koşul sınaması `while` da olduğu gibi döngüye girmeden yapılır. Bu döngü deyiminde içinde

diğerlerinden farklı olarak başlangıç değeri ve döngü sayacına sahip olmasıdır. Bu deyimin kullanımı Program 7.3 de gösterilmiştir Genel yazım biçimi:

```
for( başlangıç ; koşul ; artım )
{
    3
    Koşula bağlı olarak yapılacak işlemler;
}
```

İç içe Geçmiş Döngüler

Bir program içinde birbiri içine geçmiş birden çok döngü de kullanılabilir. Bu durumda (bütün programlama dillerinde olduğu gibi) önce içteki döngü, daha sonra dıştaki döngü icra edilir.

```
for(a=1; a<=9; a++)
    for(b=0; b<=9; b++)
        for(c=0; c<=9; c++)
        {
            sayi = 100*a + 10*b + c;          /* sayi = abc (üç basamaklı) */
            kup  = a*a*a + b*b*b + c*c*c;    /* kup  = a^3+b^3+c^3          */

            if( sayi==kup ) printf("%d. %d\n",k++,sayi);
        }
```

break Deyimi

Bir C programında, bir işlem gerçekleştirilirken, işlemin sona erdirilmesi bu deyim ile yapılır. Örneğin, döngü deyimleri içindekiler yürütülürken, çevrimin, koşuldan bağımsız kesin olarak sonlanması gerektiğinde bu deyim kullanılır. Mesela:

```
...
do{
    scanf ("%d", &x);

    if(x==0) break;

    printf("%f", 1.0/x);

}while(1);
```

continue Deyimi

Bir döngü içerisinde continue deyimi ile karşılaşırsa, ondan sonra gelen deyimler atlanır ve döngü bir sonraki çevrime girer. Örneğin:

```
for(x=-50; i<=50; x++)
{
    if(x<0) continue;          /* x<0 ise alttaki satırı atla */
    printf("%d\t%f", x, sqrt(x));
```

KAYNAKLAR

- 1- Programlamaya Giriş ve Algoritmalar Ders Notları - Revizyon ve Baskı Bilgisi
69/17.10.2007 01:51:28 PM
- 2- Algoritma ve Programlamaya Giriş Ders İçerikleri - Yrd. Doç. Dr. Nurşen SUÇSUZ,
Trakya Üniversitesi Tunca Meslek Yüksekokulu, Edirne-2009
- 3- <http://www1.gantep.edu.tr/~bingul/c/index.php?ders=1> , Dr. Ahmet Bingül, Gaziantep Üniversitesi , Fizik Mühendisliği Bölümü , Nisan 2011