T.C.
MİLLÎ EĞİTİM BAKANLIĞI

ENDÜSTRİYEL OTOMASYON TEKNOLOJİLERİ

PROGRAMLAMAYA GİRİŞ

Ankara, 2014

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR

ALAN	**Endüstriyel Otomasyon Teknolojileri**
DAL/MESLEK	**Ortak Alan**
MODÜLÜN ADI	**Programlamaya Giriş**
MODÜLÜN TANIMI	Bilgisayar programı yazım tekniklerini anlatan öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	
YETERLİK	Bu modülü tamamladığınızda bilgisayar programı yazabileceksiniz.
MODÜLÜN AMACI	**Genel Amaç** Bilgisayar programını programlama dili tekniklerine uygun olarak yazabileceksiniz. **Amaçlar** **1.** Bilgisayar programlamada ekrana yazdırma ve karakter alma komutlarının yazımını doğru bir şekilde yapabileceksiniz. **2.** Bilgisayar programlamada karar yapılarını doğru bir şekilde kullanabileceksiniz. **3.** Bilgisayar programlamada döngü yapılarını doğru bir şekilde kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam:** Programlama Laboratuarı **Donanım:** Bilgisayar, Bilgisayar çevre birimleri, programlama akış şablonu
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen, modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma, eşleştirme vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili öğrenci,

Bu modülden itibaren C programlama dilini kullanmaya başlayacaksınız. C programlama dilini sadece program yaparken değil, mikroişlemciler ve mikrodenetleyiciler ile de kullanabilirsiniz. Yaygın olarak kullanılan programlama dillerinden C++, Java ve birçok script dili C rotasyonunu veya çok benzer bir yapıyı kullanır. Bu yüzden burada öğrenilecek konular diğer programlama dillerinin öğrenilmesinde temel olacaktır.

Programlamaya Giriş modülü, 3 öğrenme faaliyetinden oluşmaktadır.

C programlama temelleri
Karar yapıları
Döngüler

Öğrenme faaliyetlerinde konu genel olarak değil, örnekler üzerinde anlatılmıştır. Bu yöntem, yapacağınız uygulamalara rehberlik edecektir. Örneklerde yazılacak programın ekran görüntüsü, akış diyagramı, değişken tablosu, örnek program ve açıklamaları verilmiştir. Bu yol ile uygulamalarda yararlanacağınız işlem basamaklarını daha iyi anlayacaksınız.

Programlamayı öğrenme aşamasında bazı zorluklarla karşılaşacaksınız. Bu zorluklar karşısında sorunu teşhis edip, çözümü kendiniz düşünmelisiniz. İlk anlarda yaşayacağınız sorunlar belirli bir aşama sonunda kolay bir hâle gelecektir.

Programlamanın en önemli kısmı algoritma hazırlayabilmektir. Algoritma hazırlandıktan sonra hazırlanan algoritmanın herhangi bir programlama dilinde kodlanması işin en basit kısmıdır. Burada önemli olan programlama dili değil problemin çözümü için algoritma geliştirebilmektir.

Bu nedenle programlamaya hazırlık modülünün ilk öğrenme faaliyetinde algoritma oluşturmayı öğreneceksiniz. İkinci öğrenme faaliyetinde ise bilgisayara ilk programınızı yazacak ve program çıktısını ekran üzerine göreceksiniz.

Bilgisayarı kendi amacınız için programlamak ve programınızın sonucunu görmenin çalışmalarınızda motivasyon sağlayacaktır.

ÖĞRENME FAALİYETİ -1

AMAÇ

Bilgisayar programlamada veriyi ekrana yazdırma ve klavyeden karakter alma komutlarının yazımını doğru bir şekilde yapabileceksiniz.

ARAŞTIRMA

- C programlama dili kuralları hakkında araştırma yapınız.

1. C PROGRAMLAMA TEMELLERİ

AT&T Bell Laboratuarlarında , Ken Thompson ve Dennis Ritchie tarafından UNIX işletim sistemini geliştirebilmek amacıyla B dilinden türetilmiş yapısal bir programlama dilidir. Geliştirilme tarihi 1972 olmasına rağmen yayılıp yaygınlaşması Brian Kernighan ve Dennis M. Ritchie tarafından yayımlanan "C Programlama Dili" kitabından sonra hızlanmıştır. Günümüzde neredeyse tüm işletim sistemlerinin (Microsoft Windows, GNU/Linux, *BSD, Minix) yapımında %95' lere varan oranda kullanılmış, halen daha sistem, sürücü yazılımı, işletim sistemi modülleri ve hız gereken her yerde kullanılan oldukça yaygın ve sınırları belirsiz oldukça keskin bir dildir.

1.1. C Program Yapısı

Aşağıdaki program C dilinde en temel yapıya bir örnek olarak gösterilebilir.

Örnek:
"Nasılsınız?
Güzel bir gün" ifadelerini görüntüleyiniz.

Akış diyagramı

Program

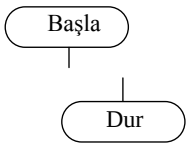
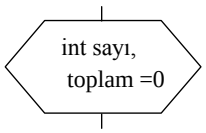
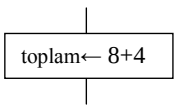
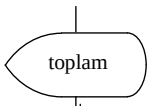
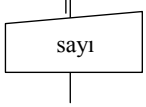
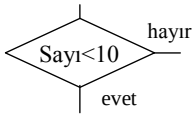
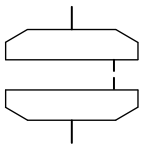
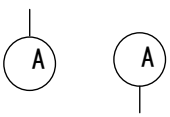
```
/* Örnek 1.1 */                                  //--- 1

#include<stdio.h>                               //--- 2

main()                                          //--- 3
{
      printf("Nasılsınız ?\n");                 //--- 4
      printf("Güzel bir gün\n");
}
```

Ekran görüntüsü

```
Nasılsınız?
Güzel bir gün.
```

Program Açıklaması

Bilgisayar programlamada kullanılan genel akış sembolleri şekil 1.1'de listelenmiştir. Algoritmalar, bu semboller ile oluşturulduktan sonra program koduna çevrilir.

İSİM	SEMBOL	AÇIKLAMA
Başlatma ve durdurma		Başla, akış diyagramının en üst kısmına konmalıdır. Dur ise en altta olmalıdır.
Tanımlama ve atama		Değişkenlerin tanımlanması ve başlangıç değerlerinin atanması için yapılan hazırlık.
Süreç		Bu herhangi bir süreç, fonksiyon veya işlemi ifade eder.
Çıktı		Ekranda görüntüleme
Giriş		Klavyeden veya diğer giriş aygıtlarından veri girişi
Karar verme		Karar verilme noktasındaki bağlantı noktası.
Döngü		Bir komut veya komutlar grubunun belirlenmiş şarta göre tekrarlanması için kullanılır.
Bağlantı		Bir akış diyagramından aynı akış diyagramının başka bir bölümüne geçiş yapmak için kullanılır.

Şekil 1.1 : Akış diyagramı sembolleri

1) Açıklama Satırı

Program kodlarını makine diline çeviren C dili derleyicisi /* ve */ karakterleri arasında kalan bölümleri ihmal eder. Bu yüzden programcılar bu karakterlerin arasına

programın anlaşılmasını kolaylaştıran açıklamaları yazarlar. Sadece tek bir satır ihmal edilmek istenirse // işareti kullanılabilir.

2) # İnclude İfadesi

Bir satır # işaretiyle başlıyorsa bunun manası, bir sonraki komutu (include), derleme işleminden önce dikkate al demektir. Programın başında yazılan *#include<stdio.h>* ifadesi, derleyiciye ismi stdio.h olan standart giriş-çıkış (standard input output) başlık dosyasını okumasını ister. Giriş-çıkış fonksiyonlarını kullanan bir programı derlemek için gerekli bilgiler bu dosyada tutulurlar. Her programda bir giriş-çıkış fonksiyonu kullanılmalıdır.

3) Main fonksiyonu

C programı fonksiyonlardan oluşmuştur. En basit hâliyle bir fonksiyon aşağıdaki parçalardan oluşur;

```
Fonksiyon ismi ()
{

}
```

Her C programı, main fonksiyonuna sahip olmalıdır. Programın çalıştırılması bu fonksiyon içerisinde yapılmalıdır. Bu program örneğinde main fonksiyonu satır 5'ten 9'a kadardır. Bir fonksiyon içerisinde belirli bir işlemi gerçekleştiren komutlar yer alır. Bir komut, diğerinden noktalı virgül yardımıyla ayrılır. Bu ayırma işlemi için kullanılan sembole **"sınırlayıcı"** adı verilir.

```
printf("Nasılsınız\n");
```

komut **sınırlayıcı**

4) Printf Fonksiyonu

Printf fonksiyonu, belirli bir veriyi ekranda görüntülemek için kullanılan formatlı yazım fonksiyonudur. C dili tarafından daha önceden belirlenmiş bir fonksiyon olduğu için standart bir fonksiyondur. Printf fonksiyonunda yazdırılmak istenen ifadeler " " karakterleri arasında olmalıdır. Bu ifadelere karakter katarı (string) denir. Ekranda görüntülenmek istenen ifadeden sonra kullanılan \n karakterleri ise imleci bir alt satıra geçirmek için kullanılır. Bir fonksiyonun kullanımı şu şekildedir:

```
Fonksiyon ismi ( argümanları);
```

```
printf("Nasılsınız\n");
```

fonksiyon **argüman**

Program yapısını daha iyi inceleyebilmek için kullanılan yöntemler

Boş Satırlar

Örneğimizde 2. ve 4. satırlarda herhangi bir komut yoktur. Bu boşluklar; programın gruplar hâlinde görülmesini sağlayarak, takip edilmesini kolaylaştıracaktır.

Girinti

Girinti sayısı isteğe bağlı olmak üzere ayarlanabilir. Genel olarak ana fonksiyon içerisindeki tüm komutlar 5. sütundan itibaren yazılmaya başlanır. Bu durum tüm program içerisinde sabit olmalıdır. Uzun ve karmaşık programlarda girinti sayısı artırılabilir. Her programcı, bu konuda başlangıçtan itibaren çok dikkatli olmalıdır. Bu şekilde programda oluşan hataları bulmak kolaylaşacaktır.

Örnek:

```
main()
{
        int sayi;
```

5. sütun

1.2. Değişkenler Ve Printf Fonksiyonu

Aşağıdaki örnek, C dilinde yapılan basit bir işlemi göstermektedir.

Örnek:

4 ve 8 sayılarının toplama, çıkarma, çarpma ve bölme sonuçları ekranda gösteren programı yazınız.

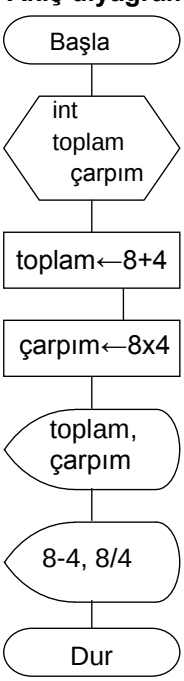
Ekran görüntüsü

```
Toplama=12 Çarpma=32
Fark=4  Bölüm=2
```

Program

```
/* Örnek 1.2 */

#include<stdio.h>

main()
{
    int toplam,carpim;                                  //--1
    toplam=8+4;                                         //--2
    carpim=8*4;                                         //--3

    printf("Toplama=%d Çarpma=%d\n",toplam,carpim);     //--4
    printf("Fark=%d Bölüm=%d \n",8-4,8/4);
}
```

Akış diyagramı

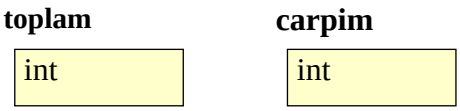


Program Açıklaması

Akış diyagramında görüldüğü gibi bu program, hesaplama sonuçlarını göstermek için iki farklı yol kullanmaktadır. Birincisinde değişkenler kullanılarak toplama ve çarpma sonuçları gösterilmiştir. Diğerinde ise çıkarma ve bölme sonuçları doğrudan printf fonksiyonunda kullanılarak gösterilmiştir.

1) int toplam,carpim

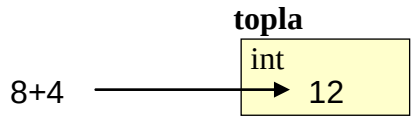
Değişkenler, program çalıştığı sürece kullanılan verilerin bellek içerisinde saklandığı verilerdir. Değişkenler her zaman fonksiyonların başlangıcında tanımlanmak zorundadır. Değişkenleri, içerisine sayıları koyabileceğimiz, üzerinde isim yazılı kutular olarak düşünebiliriz.



"**Değişken tipi**", değişken isimlerinden önce tanımlanmalıdır. Değişkenler tam sayıları tutarken int (integer) olarak tanımlanırlar. "int" tanımlamasının kullanımını örnek1-2'de 1 no.lu satırda görebilirsiniz.

2) toplam = 8 + 4

Burada 8+4'ün toplamının sonucu toplam değişkeninin içerisine atanmıştır.



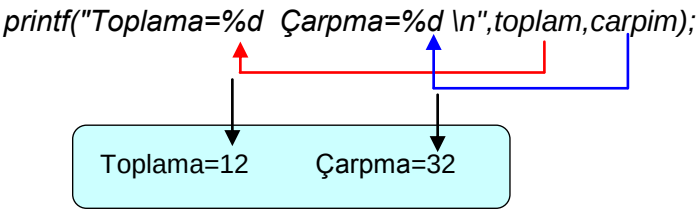
C dili dahil birçok programlama dilinde, değer atamaları " = " karakteri ile yapılmaktadır. C dilindeki " = " karakteri, matematikteki eşit anlamından farklıdır. Burada yapılan işlem atama işlemidir ve bu fark karıştırılmamalıdır. İşlemlerde, sol taraf "atanılan" sağ taraf ise "atanan"'dır.

3) carp = 8 * 4

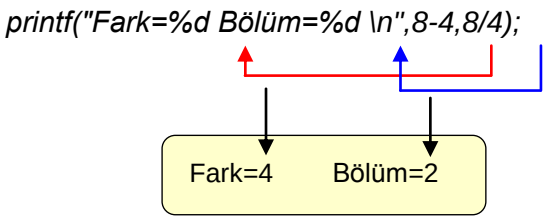
Burada, çarpma işleminin sonucu carp değişkeninin içersine atanmıştır. Programlama dillerinde çarpma işlemlerinde, "**x**" (çarpım) karakteri yerine, " * " karakteri kullanılır.

4) Tip Dönüşüm Belirteci

Printf fonksiyonunun içerisindeki tip dönüşüm belirteci, "," den sonra değişken içerisinde sayısal bir değerin kullanıldığını göstermektedir.



Printf fonksiyonu içerisindeki sayısal formülleri, aşağıda gösterildiği gibi de kullanabiliriz. Çıkarma ve bölme işlemlerini yaparken (-) ve (/) karakterleri doğrudan fonksiyonun içindeki yerine dikkat ediniz.



1.3. Değişken İsimleri ve Tipleri

Programın yazımına geçmeden öncelikle kullanılacak değişken isimleri belirlenmelidir. Değişken isimleri böcek (bug) adı verilen program hatalarını bulma işlemi (debugging) sırasında programcıya veya programı daha sonra inceleyen kişilere yardımcı olma açısından oldukça önemlidir. C dilinde kullanılan değişken isimlendirme kuraları aşağıda görülmektedir.

Değişken İsimlendirme Kuralları

1) Değişken isimlendirmesinde İngiliz alfabesinde yer alan 26 karakter, (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) rakam karakterleri (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) ve altçizgi (_) karakteri kullanılır. Değişken isimleri rakam ile başlayamaz.

2) İsimlendirmelerde yukarıda belirtilen karakterlerin dışında başka bir karakterin kullanılması derleme zamanında hata oluşumuna yol açar (ömeğin boşluk karakterinin kullanılması Türkçe karakterlerin kullanılması, +, -, /, *, & ya da $ karakterinin kullanılması gibi).

3) C dilinde isimlerin maksimum uzunluğu derleyicilere göre değişebilir. Ancak birçok derleyicide 32 sayısı kullanılmaktadır. Eğer verilen isim 32 karakterden daha fazla karakter içeriyorsa, derleyici yalnız ilk 32 karakterini algılar.

4) Değişken isimleri C programlama dilinde daha önceden belirlenmiş komutlarından (int veya printf gibi) herhangi birisi olamaz.

5) Değişken isimlerini belirlerken, karakterler genel olarak küçük harf olarak seçilir.

Bir değişkenin tipi, içerisinde saklanacak olan verinin türü ile belirlenir. Tam sayıları saklamak için **int** değişken türü kullanılır. Eğer ondalıklı sayıları tanımlamak istiyorsanız maksimum değer veya sayıların ondalık hassasiyetine göre float türünde veya double türünde bir değişken tanımlaması yapılmalıdır.

Derleyicilere göre değişen değer aralıklarını, program yazmaya başlamadan önce kontrol etmek gerekir. Aşağıdaki şekil 32 bit CPU(merkezi işlem birimi) için derleyicilere göre değişen değer aralıklarını göstermektedir.

Tip	Byte Uzunluğu	Değer Aralığı
char	1	-128 ~ 127
short	2	-32768(-2^{15}) ~32767 (2^{15} -1)
int	4	-2147483648 (-2^{31}) ~2147483647 (-2^{31}-1)
long	4	-2147483648 (-2^{31}) ~2147483647 (-2^{31}-1)
float	4	$\pm 10^{-37}$ ~$10^{38.}$ Hassasiyet 6 dijit.
double	8	$\pm 10^{-307}$~$10^{308.}$ Hassasiyet 15 dijit.

Şekil 1.2: Değişken tipleri

1.4. Printf Fonksiyonu

Genel Kullanım

printf(“*string*”,**değişken** veya **değerler**)

Eğer birden fazla kullanılacaksa “,” ile ayrılmaları gereklidir.

Kullanım 1)

Program

```
#include<stdio.h>
main()
{
    printf("'A\' Bir karakterdir.\n'1\'de karakterdir .\n");
}
```

Ekran görüntüsü

```
‘A’ bir karakterdir.
‘1’ de karakterdir.
```

Printf( ) fonksiyonu “ “ (çift tırnak) karakterleri arasında kalan argümanları göstermek için kullanılan bir fonksiyondur. Bu fonksiyonu kullanırken karakterlerin arasına \ , ‘ , ” gibi karakterleri doğrudan kullanamayız. Bu karakterleri kullanabilmek için daha önce \

karakterini kullanmamız gerekir. Şekil 1.3'te gösterilen "çıktı kodu" diye tanımlanan karakterler ve \ karakteri kontrol amaçlı kullanılan karakterlerdir. Bu karakterler yukarıda görüldüğü gibi, imlecin konumunun belirlenmesi gibi çok sık görülen işlemlerde kullanılırlar.

Çıktı kodu	Fonksiyon
\n	İmleci bir sonraki satırın başına konumlandırır.
\t	İmleci bir sonraki tab konumuna götürür.
\a	Bilgisayarın zilini çalar.
\"	" çift tırnak işaretini gösterir.
\'	' tırnak işaretini gösterir.
\\	\ karakterini gösterir.

Şekil 1.3: Çıktı kodu karakterleri

Kullanım 2)

Program

```
#include<stdio.h>
main()
{
      int toplam;
      float ort;

      toplam=41;
      ort=toplam/4.0;
      printf("ToplamSonuç= %d. ortalama= %f. \n", toplam,ort);
}
```

Ekran görüntüsü

```
ToplamSonuç=41.  Ortalama= 10.25.
```

Bu örnekte printf() fonksiyonu argüman içerisindeki bir string (karakter katarı) ifadeyi takip eden değişkenlerin içeriklerini gösterir. Her zaman argümanlar "**,**" ile birbirlerinden ayrılırlar. %d ve %f tip dönüşüm belirtecidir. Bir karakter katarının yeri ve kullanılış biçimi aşağıdaki şekildedir:

```
printf("ToplamSonuç= %d  Ortalama= %f \n",toplam,ort);
```

Desimal sayı

Gerçel (ondalık) sayı

Şekil 1.4 sık kullanılan tip dönüşüm belirteçlerinin listesini göstermektedir.

Tip Dönüşüm Belirteçleri	Fonksiyon
%d	Tamsayıları gösterir
%ld	Uzun tamsayıları gösterir.
%x , %X	Onaltı tabanlı (hekzadesimal) sayıları gösterir. (%x a' dan f' ye kadar olan küçük karakterleri kullanır. %X ise A'dan F'ye kadar olan büyük harfleri kullanır.)
%f	Ondalıklı sayıları gösterir.
%c	Kodu karaktere dönüştürür.
%s	Karakter katarını (string) gösterir.
%p	Adresi gösterir.
%%	% karakterini gösterir.

Şekil 1.4: Tip dönüşüm belirteçleri

Kullanım 3)

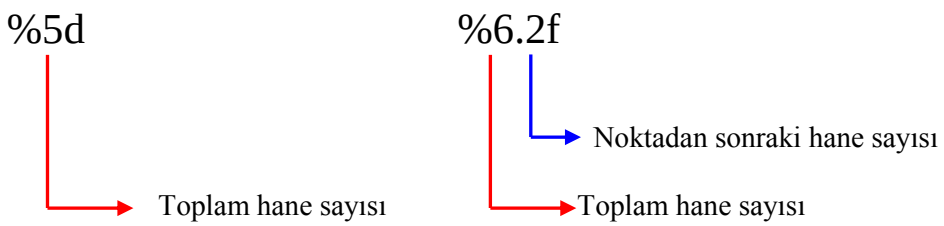
Printf fonksiyonundaki hane (dijit) sayısını belirleyebiliriz.

```
printf("ToplamSonuç= %5d Ortalama= %6.2fn", toplam, ort);
```

Ekran görüntüsü

ToplamSonuç = ـــ200 Ortalama = ـ66.67

Bu programda tip dönüşüm belirteciyle birlikte kullanılan sayılar çıktının formatını belirlemektedir. Bu sayılar ayrılan alanın genişliğini ayarlamak için kullanılır.



Ayrılan alan genişliği ile ilgili açıklamalar

1) Sayısal kısmın tamamı, belirtilen toplam basamak sayısını aştığı zaman toplam basamak tanımlaması ihmal edilir.

2) Desimal olan bölümler için noktadan sonraki sayıya göre yuvarlama işlemi yapılır.

%6.2f 66.6666

Yuvarlama 66.67

3) Gerçel sayının tam kısmı eğer belirtilmeyecekse %.2f şeklinde de sadece ondalıklı kısım için yer ayrılabilir kısmının genişliği basamak sayısı kaç ise o olacaktır.

Örnek :

Printf fonksiyonunun kullanılışı ile ilgili olarak aşağıdaki kodların çıktısı ne olur? A'nın değeri 15, b'nin değeri 1240.48 olarak farz edilecektir.

```
(1) printf("cevap=%5d ,%10.1f \n",a,b)
(2) printf("cevap=%4d cevap(hex)=%2X \n",a,a);
(3) printf("cevap=%2.1f\n",b);
(4) printf("cevap=\\%.2f \n",b);
```

Açıklama

1. ␣␣␣15 ➔ a değişkenine 5 basamak ayrılmıştır.

␣␣␣␣␣1240.5 ➔ b değişkenine 10 basamak ayrılmıştır. Ondalıklı kısımdan sonra bir basamak olduğu için sadece 48 değeri 5 değerine yuvarlanmıştır.

2. ␣␣15 ➔ a değişkenine 4 basamak ayrılmıştır.

␣F ➔ a değişkenine 2 basamak ayrılmıştır ve sonuç hegzadesimaldir.

3. 1240.5 ➔ b değişkeni için ayrılan basamak değeri, b değerinden küçük olduğu için tam sayı değeri aynı şekilde görünür; fakat ondalıklı kısım için tek basamak ayrılmasından dolayı sadece 5 rakamını görebiliriz.

4. \1240.48 ➔ b değişkeninde ondalıklı kısımda 2 basamak ayrıldığı için sayının tamamı görünür. Sayının solundaki \\ karakterleri \ karakterini gösterir.

1.5. Scanf Fonksiyonu

scanf fonksiyonu klavyeden veri girişi yapmak için kullanılır.

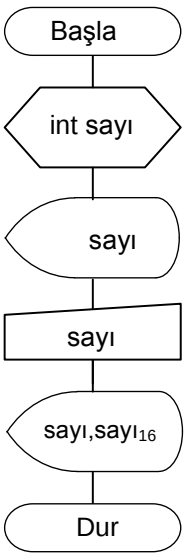
Örnek :

Klavyeden girilen onluk tabanda (desimal) bir sayıyı on altılık tabana(hekzadesimal) dönüştüren programı yazınız.

Ekran görüntüsü

```
Onluk sistemde bir sayi giriniz ==>242
242 sayisi hekzadesimal F2 sayisina eşittir.
```

Akış diyagramı



Program

```
/* Örnek 1.4 */


#include<stdio.h>


main()

{
   int sayi;



   printf("Onluk sistemde bir sayi giriniz==>");          //---1
   scanf("%d",&sayi);                                     //---2

   printf("%d sayısı hekzadecimal %x sayısına eşittir",
   sayi,sayi);                                            //---3
}
```

Program Açıklaması

1) Kullanıcıya onluk sistemde bir sayı girmesi gerektiğini söyleyen ifadenin ekranda görüntülenmesi.

2) **scanf** fonksiyonuyla girilen değerin sayı değişkeninin içinde tutulması. Tip dönüşüm belirteci %d, girilen değerin onluk sistemde olduğunu ve sayı değişkeniyle ifade edildiğini gösterir. Değişkenden önce kullanılan & karakteri, adres operatörü olarak isimlendirilir ve scanf fonksiyonunun kullanımında değişken içerisindeki değerin değişimi için gereklidir. Printf fonksiyonu kullanıldığında, printf sadece değişkenin içerisindeki değeri kullanır; değişkenin içeriğini değiştirmez.

3) **Printf** fonksiyonu sayı değişkeninin içerisindeki %x tip dönüşüm belirteciyle üretilen 16'lık sistemdeki karşılığını gösterir.

Genel kullanım

scanf(”*string*”,&değişken ismi)
eğer birden fazla değişken kullanılacaksa, birbirinden “,” ile ayrılması gereklidir.

Şekil 1.4'te görüldüğü gibi tip dönüşüm belirtecinin scanf fonksiyonundaki kullanımı, printf fonksiyonu kullanımının aynısıdır. %d onluk sayıları ,%f gerçel sayıları, %X veya %x 16'lık sayı sistemindeki sayıları ifade etmek için kullanılır. Alan genişliğini scanf fonksiyonunda belirleme şansımız yoktur.

Aşağıdaki program örneğinde, bir scanf fonksiyonuyla birden fazla sayıda değer girebiliyoruz. Bu durumda klavyeden girilen değerler, boşluk veya boşluklarla ayrılmalıdır. Aynı değerde girilen iki sayının %d ve %x belirteçlerinden dolayı farklı yorumlandığına dikkat ediniz.

Program

```
/* Ornek Program */

#include<stdio.h>
main()
{
      int sayi1,sayi2;

      printf("Onluk ve Onaltilik sistemde sayi giriniz ===>");
      scanf("%d %x",&sayi1,&sayi2);

      printf("Girilen sayilar %d ve %d.",sayi1,sayi2);
}
```

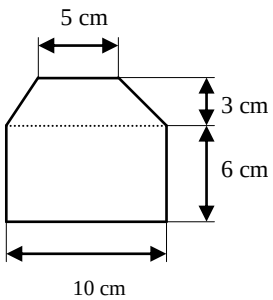
Ekran görüntüsü

```
Onluk sistemde ve Onaltilik sistemde sayi giriniz ===>10 10
Girilen sayilar 10 ve 16
```

1.6. Formül İfadeleri

C programlama dilinde matematiksel hesaplamaları yapmak için kullanılan bazı kurallar vardır. Bu kurallar aşağıdaki örnekte görülmektedir:

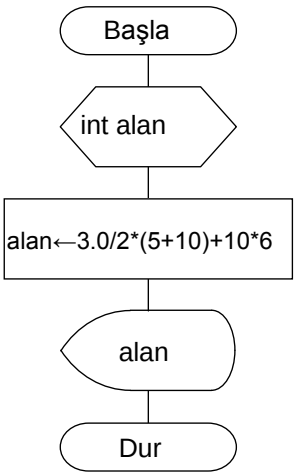
Örnek:
Aşağıda verilen şeklin alanını aşağıdaki ekran görüntüsündeki gibi görüntüleyen programı yazınız.



Ekran görüntüsü

```
Şeklin alanı= 82 [cm·cm]
```

Akış diyagramı



Program

```
/* ornek 1.5 */

#include<stdio.h>
main()
{
        int alan;
        alan = 3.0/2*(5+10)+10*6;
        printf("Seklin Alanı=%d [cm·cm] \n",alan);
}
```

Program Açıklaması

Programdaki şeklin alan hesaplamasında yüksekliğin 3 yerine 3.0 olarak alındığına dikkat edin. Bu durum aşağıda açıklanmıştır:

1) Sayısal değerler

Programlamada kullanılan sayısal değerler şekil 2.4'de olduğu gibi sekizlik veya on altılık sistemde olabilir.

alan = 3.0/2*(5+10)+10*6

	Açıklama	Örnek	tip
Tam Sayı	0 (sıfır) hariç, onluk sistemdeki bir sayı	16,102345	int
	0 ile başlayan 8'lik sistemde bir sayı	020	
	16'lık sistemde 0x veya 0X ile başlayan bir sayı	0x10,0xfb3 0X10,0XFB3	
Gerçel Sayı	(Noktalı sayılar)	5.5,.125	double

Şekil 1.5: Sayısal değerler

2) İşlem Önceliği

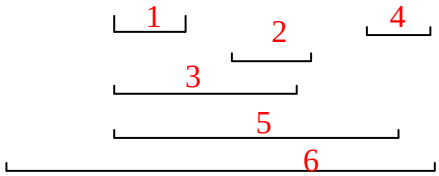
+ , - , * , / karakterleri **operatör** olarak isimlendirilirler. Şekil 1.6'da operatörlerin işlem öncelikleri gösterilmiştir. Aynı önceliğe sahip operatörler varsa işlem belirli bir yönde yapılır. ( ) işaretleri, bir operatör değildir; fakat aralarında kalan işlemler en yüksek önceliğe sahiptir.

Öncelik		Operatör	İşlem Yönü
Yüksek ↑	1	*(Çarpma) / (Bölme) %(kalan, mod)	(Sol) → (Sağ)
	2	+(toplama) – (çıkarma)	→
↓ Alçak	3	= (Eşittir)	←

Şekil 1.6: Operatör öncelikleri

Örnekte kullanılan ifadenin işlem sırası aşağıda gösterilmiştir.

alan=3.0/2 * (5+10) + 10*6



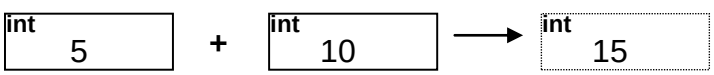
Aktarma "=" en düşük önceliğe sahip olduğu için bu işlem en son yapılacaktır. Çarpma ve bölme en yüksek önceliğe sahip olduğu için ilk olarak soldan sağa doğru bu işlemler işletilecektir. Bu yüzden bölme ilk sırada, toplama parantezden ( ) dolayı ikinci sırada ve çarpma üçüncü sırada olmak üzere işlemler yapılacaktır. Dördüncü sıradaki çarpma işlemi yapılacak ve en düşük önceliğe sahip toplama işlemi bitirildikten sonra değer aktarımı gerçekleşecektir.

3) İşlem Kuralı

Operatörler kullanılarak yapılan işlemlerin açıklanması:

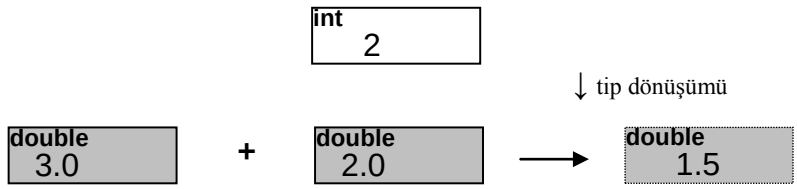
a) Aynı Tip Değerlerin İşlenmesi

5 + 10 değerlerinin toplanması için 5 ve 10 tipleri integer olan sayısal sabitlerdir. Bu durumda bilgisayar, oluşacak sonuç için aynı tipte bir depolama yeri hazırlar.

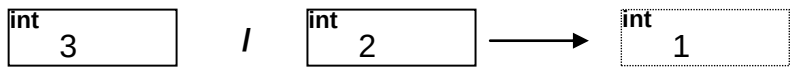


b) Karma İşlemi

Farklı iki tipteki değerin işlenmesine karma işlem (hibrid işlem) adı verilir. Bir karma işlemde int ve double tipindeki değerler otomatik olarak double tipinde farz edilir. Sonuç olarak elde edilen sayı da double tipinde olacaktır.



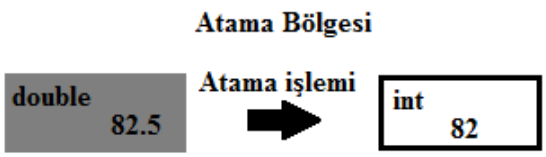
Bu şekilde elde edilen sonuç gerçel sayı olacaktır.3.0 yerine 3 yazdığımızda daha önceden bahsedilen kural gereği geçici saklama bölgesinde ayrılan yer integer olacak ve sonuçta 1.5 yerine 1 olarak hesaplanacaktır.



Hesaplamalarda bu durumu sürekli göz önünde bulundurmalıyız.

c) Farklı Tipteki Değişkenlerin Atanması

Sağ taraftaki işlemin sonucu son olarak gerçel sayı 82.5 olur. "=" integer tipindeki değişken alanının yerine kullanılır. Bu durumda noktadan sonraki sayı ihmal edilir.



Böylece değişken alanın değeri 82 olur. Yukarıdaki örnek, sayısal sabitleri anlatan veya değişken işlemlerini açıklayan bir örnektir.

1.7. Tip Dönüşüm Operatörleri (Cast operatörü)

Yukarıdaki kurala göre iki integer sayının bölümünden elde edilen sonuçta ondalıklı kısım bulunmayacaktır. Bir sonraki programda kullanılan %6.1f şeklinde kullanılan dönüşüm belirtecinin kullanılmasında dahi ondalıklı sayı elde edilmez.

Program

```
/* Örnek Program */
#include<stdio.h>
main()
{
        int toplam, sayi;
        printf("Toplam agirlik ve  ögrenci sayisini giriniz===>");
        scanf("%d%d",&toplam,&sayi);

        printf("Agirlik ortalama=%6.1f  \ n",toplam/sayi);              //---1
}
```

Bu durumda, 1 no`lu satırı şu şekilde düzenleyerek sonucun gerçel sayı olarak elde edilmesini sağlayabiliriz.

printf("Ortalama Ağırlık=%6.1f\n", (float)toplam/sayi);

(float)toplam komutunu eklediğimizde **toplam** değişkeni sadece bu hesaplama için float tipinde bir değişken hâline gelecektir. Böylece parantez içindeki veri tipi geçici olarak değişmiş olacaktır. Bu tür operatörlere **tip dönüşüm operatörleri** adı verilir. Buna göre (float)toplam/sayi hesaplamasına **karma (hybrid) hesaplama** adı verilir. Sonuç bizim istediğimiz gibi float türünde olacaktır.

1.8. Modül Operatörü

Modül operatörü, iki integer değerin bölümünden kalanı verir. Bir sonraki program**,** iki sayının bölümünden kalanı veren program örneğidir.

Program

```
#include<stdio.h>

main()
{
        int sayi1=8,sayi2=5;                                                    //---1
        printf("%d / %d = %d kalan %d\n",sayi1,sayi2,sayi1/sayi2,sayi1%sayi2);  //---2
}
```

Ekran görüntüsü

```
8 / 5 = 1 kalan 3
```

[İleri Çalışma] Matematiksel fonksiyonların kullanımı

Aşağıdaki matematiksel fonksiyonlar C dilinde kullanılan fonksiyonlardır..

fonksiyon	Açıklamalar
sin(x)	Radyan cinsinden verilen X'in sinüsünü bulur.
cos(x)	Radyan cinsinden verilen X'in kosinüsünü bulur.
tan(x)	Radyan cinsinden verilen X'in tanjantını bulur.
asin(x)	arc sin
acos(x)	arc cos
atan(x)	arc tanjant
exp(x)	e^x
log(x)	e tabanına göre logaritmayı bulur (x > 0 olmalıdır).
log10(x)	10 tabanında logaritma (x > 0 olmalıdır).
pow(x,y)	x^y
Sqrt(x)	Karekök alır(x >= olmalıdır).
fabs(x)	Mutlak değer

Matematiksel fonksiyonları kullandığımızda aşağıdaki kütüphaneyi eklememiz gerekir:

#include<math.h>

UYGULAMA FAALİYETİ

Aşağıdaki sorulara ilişkin uygulama faaliyetini yapınız.

- Aşağıdaki ekran görüntüsünde rakamların belirtilen genişlikte yazdırılmasını sağlayacak programı yazınız.

```
Uzunluk=␣15[cm]  Genişlik =␣20[cm]
Yükseklik=␣ 5[cm]
Ağırlık=␣2.4[kg]  Sayı= ␣␣12
```

- Klavyeden girilen 3 tam sayının toplamını ekranda gösteren programı yazınız.

- Bir üçgenin tabanını, yüksekliğini gerçel sayılar olarak klavyeden kabul eden ve alanını hesaplayarak ekranda gösteren programı yazınız.

- Klavyeden girilen 3 adet paralel direnç değerine göre toplam direnç değerini hesaplayan programı yazınız (direnç değerleri tam sayı, toplam sonuç değeri ise gerçel olacak).

- Toplam saniyeleri kabul ederek bunları saat, dakika ve saniye cinsinden gösteren programı yazınız.

```
saniyeleri giriniz ==> 3610
1 saat 0 dakika ve 10 saniye
```

İşlem Basamakları	Öneriler
➢ Değişken tablosunu hazırlayınız.	➢ Programda kullanacağınız değişkenlerin tipini belirleyiniz. ➢ Değişken isimlendirme kurallarına dikkat ediniz.
➢ Akış diyagramını çiziniz.	➢ Akış diyagramı sembollerinden yararlanınız.
➢ Programı yazınız.	➢ Program satırlarının düzenli olmasına özen gösteriniz. ➢ Matematiksel işlem önceliklerine dikkat ediniz.
➢ Yazdığınız programı derleyiniz.	
➢ Programda hata var ise bunları gideriniz.	
➢ Ekran görüntüsünü kontrol ediniz.	

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri **Evet**, kazanamadığınız becerileri **Hayır** kutucuğuna (**X**) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. #include <stdio.h> kütüphanesini kullanabildiniz mi?		
2. Tip dönüşümü yapabildiniz mi?		
3. Türkçe karakter farklılıklarını gördünüz mü?		
4. Bilgi girişini yapabildiniz mi? (scanf)		
5. Bilgi çıkışı yapabildiniz mi? (printf)		
6. Yapılan matematiksel işlemlerin sonuçlarını kontrol ettiniz mi?		
7. Matematik kütüphanesini kullanabildiniz mi?		
8. Teknolojik kurallara uygun bir çalışma gerçekleştirdiniz mi?		
9. Süreyi iyi kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda "Hayır" şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız "Evet" ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi bir tam sayı değişkeninin doğru olarak yazdırılması için gereken ifadedir?
A) printf("%s", toplam);
B) print("%f", toplam);
C) printf("%d", toplam);
D) printf("%d", &toplam);

2. Aşağıdakilerden hangisi "Merhaba" ifadesini yazdırdıktan sonra yeni bir satıra geçmeye yarayan ifadedir?
A) printf("Merhaba\n");
B) printf(Merhaba, '\n');
C) printf(Merhaba\n);
D) printf('Merhaba', '\n');

3. Aşağıdakilerden hangisi *indirim* float değişkeninin değerini yazdırmaya yarar?
A) printf ("%s", indirim);
B) printf ('indirim');
C) printf ("%f", İndirim);
D) printf ("%d", indirim);

4. Aşağıdakilerden hangisi klavyeden bir desimal sayı okumak için kullanılabilir?
A) scanf("%D", &toplam);
B) scanf(toplam);
C) scanf("%S", toplam);
D) scanf("%F", &toplam);

5. Aşağıdakilerden hangisi float tipindeki bir veriyi okuyarak *indirim_oranı* isimli değişkene atar?
A) scanf("%f", indirim_oranı);
B) scanf("%d", &indirim_oranı);
C) scanf(indirim_oranı);
D) scanf("%f", &indirim_oranı);

6. Aşağıdakilerden hangisi kilo float değişkeninin değerini 2 haneli ondalıklı kısım kullanarak ekrana yazar?
A) a) printf("%f", kilo);
B) b) printf("%.2f", kilo);
C) c) printf("%2f", kilo);
D) d) printf("%f", &kilo);

Aşağıdaki soruları verilen değişken atamaları doğrultusunda yapınız.

Not: float değerini ekrana yazım formatı "%4.2f" olacak
int a=3 , b=5, c; ***float f;***

7. f=3/5+5/3 f değeri nedir?
A) 1
B) 1.00
C) 0
D) 0.10

8. c=3/5+5/3 c değeri nedir?
A) 1
B) 1.0
C) 0
D) 0.1

9. f=1/(1/100.0+1/200.0) f değeri nedir?
A) 33.34
B) 33.33
C) 66.67
D) 66.66

10. f=a/b+b/a f değeri nedir ?
A) 0.00
B) 1.01
C) 1.00
D) 1

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

Bilgisayar programlamada karar yapılarını doğru bir şekilde kullanabileceksiniz.

ARAŞTIRMA

- C programlamada karar yapıları konusu hakkında araştırma yapınız.

2. KARAR YAPILARI

Bir önceki öğrenme faaliyetinde, tüm durumların bir kere ve sıra ile çalıştırıldığı programları gördük. Bu öğrenme faaliyetinde, farklı şartlar için programın akışına yön veren karar yapılarını öğreneceksiniz.

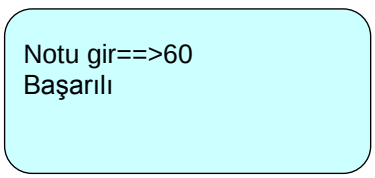
2.1. İki Yollu Karar Yapısı

Aşağıdaki program, klavyeden girilen sayı değerine göre farklı kelimeleri ekranda gösterir.

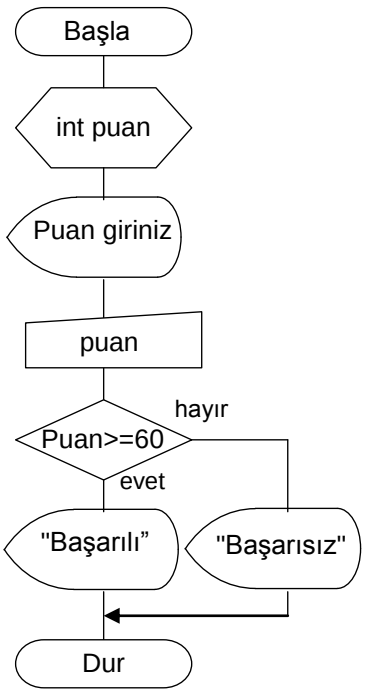
Örnek:

Ekrana 60 ve üzerinde olan puanlara göre başarılı; düşük değerlere göre de başarısız yazan programı yazınız.

Ekran görüntüsü



Akış diyagramı



Program

```
/* Örnek 2.1 */
#include<stdio.h>

main()
{
    int puan;

    printf("Puan giriniz==>");
    scanf("%d",&puan);

    if (puan >= 60)
    {
        printf("Basarılı\n");
    }
    else
    {
        printf("Basarısız\n");
    }
}
```

1

2

Program Açıklaması

Öncelikle integer tipinde puan adında bir değişken tanımlayalım. Akış diyagramı, puan değişkeninin farklı değerleri için farklı yönleri gösterecektir. Bu işlem C dilinde if komutuyla gerçekleştirilir. Programda, 1 no'lu bölüm if bloğu olarak, 2 no'lu bölüm ise else bloğu olarak isimlendirilir. Bloğun içerisinde sadece bir tane komut olduğunda küme parantezlerini { } yazmak zorunlu değildir.

```
if (puan >= 60)
        printf("Basarili\n");
  else
        printf("Basarisiz\n");
```

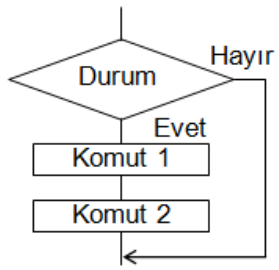
If yapısı, şartı kontrol ederek şart doğru olduğunda if bloğu içerisindeki komutlar işletilir. Eğer şart yanlış ise else bloğundaki komutlar işletilir. Şartı tanımlamak için eşitlik operatörleri ve ilişkisel operatörler kullanılır. Bu operatörler, matematiksel operatörlerden daha düşük önceliğe sahiptir.

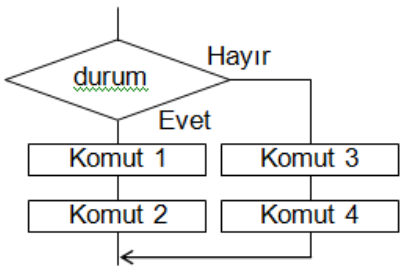
Eşitlik operatörleri	== !=	İki tarafta eşit olduğunda doğru İki tarafta eşit olmadığında
İlişkisel operatörler	> < >= <=	Sol taraf sağ taraftan daha büyük olduğunda doğru Sağ taraf sol taraftan daha büyük olduğunda doğru Sol taraf sağ tarafa eşit veya büyük olduğunda doğru Sağ taraf sol tarafa eşit veya büyük olduğunda doğru

Şekil 2.1: İlişkisel ve eşitlik operatörleri

İki yönlü karar yapısının kullanım şekilleri aşağıdaki tabloda gösterilmiştir.

Akış Diyagramı	C dili kullanımı

	if(Durum)
	{
	Komut1;
	Komut2;
	}

	if(Durum)
	{
	Komut1;
	Komut2;
	}
	else
	{
	Komut3;
	Komut4;
	}

Şekil 2.2: İki yönlü karar yapısı

Karar yapısında kullanılan mantık operatörleri aşağıdaki tabloda gösterilmiştir. Mantık operatörleri ile birden fazla durumu karar yapısında tanımlayabiliriz.

a ve b değişkenlerinin değerleri 80'in üzerinde olduğu durumlarda

```
if ( a >= 80 && b>=80){....
```

a ve b değişkenlerinden en az bir değişkenin değerinin 80 veya üzerinde olduğu durumlarda

```
if ( a >= 80 || b>=80){.....
```

Değişkenin değeri 80'den küçük olmadığı durumlarda

```
if ( !(a<80)){.....
```

Operatör	Fonksiyon	Açıklama
Şart 1 **&&** Şart 2	AND (ve)	İki şartta doğru olduğunda
Şart 1 || Şart2	OR (veya)	En az bir şart doğru olduğunda
!(şart**)**	NOT (değil)	Şart yanlış olduğunda

Şekil 2.3: Mantık operatörleri

2.2. İç İçe Karar Yapısı

Örnek:

Gönderilen malzemenin cinsine ve ağırlığına göre posta ücretini hesaplayan programı yazınız. Aşağıdaki ekran çıktısında görüldüğü gibi düzenli ve düzensiz olmak üzere iki çeşit posta çeşidi vardır. Klavyeden girilen değere göre aşağıda verilen ücretlere göre toplam ücret hesaplatılacaktır.

düzenli (25g altı) 0.6 YTL
düzensiz (50g altı) 1.2 YTL
düzenli (26 – 50g) 0.7 YTL

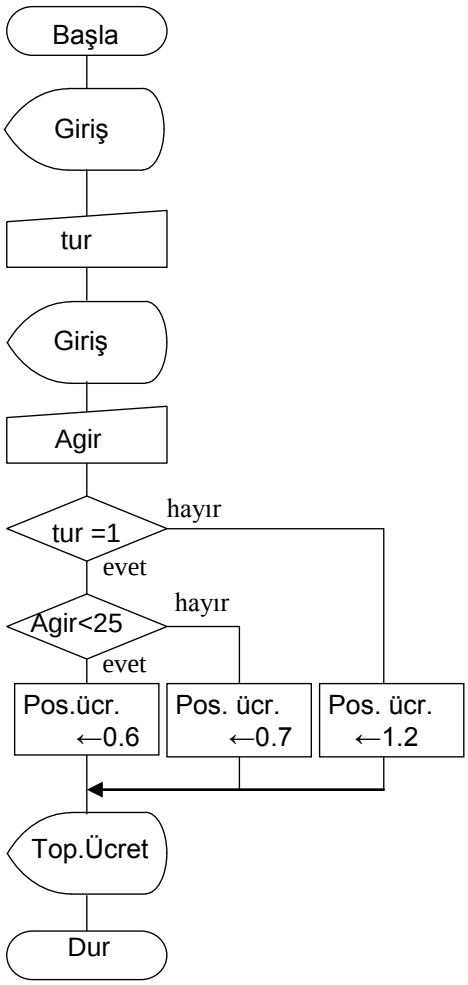
Değişken	Tip	Kullanım
Tip	int	Postanın türü
Ağırlık	int	Postanın ağırlığı
Posta ücreti	int	Postanın fiyatı

Değişkenleri tabloda görüldüğü gibi tanımlayabiliriz. Bundan sonra akış diyagramlarında değişken tanımlamak yerine, bunun gibi değişken tabloları kullanacağız.

Ekran görüntüsü

```
Tür (1.Düzenli 2.Düzensiz)==>1
Ağırlık(g)                 ==>20
Posta ucreti toplam  0.6 YTL
```

Akış diyagramı



Program

```
/* Örnek 2.2 */

#include <stdio.h>

main()
{
   int tur,agir;
   float posta;
   printf("Tür(1.Duzenli 2.Duzensiz)==>");
   scanf("%d",&tur);
   printf("Ağırlık(g)==>");
   scanf("%d",&agir);
   if ( tur ==1){
        if (agir <= 25)
          posta=0.6;
        else
        posta=0.7;
   }
   else
        posta=1.2;

printf("Posta ucreti toplam %3.1f YTL\n", posta);
}
```

Program Açıklaması

Posta türünün ve ağırlığının scanf fonksiyonuyla klavyeden girilmesi sağlandıktan sonra program öncelikle girilen ağırlığın düzenli mi düzensiz mi olduğunu kontrol eder. Düzenli olduğunda, tekrar bir if bloğu ile 25 gramın altında olup olmadığı kontrol edilir. Eğer bir if bloğunda başka bir if bloğu var ise, buna iç içe geçmiş if bloğu adı verilir.

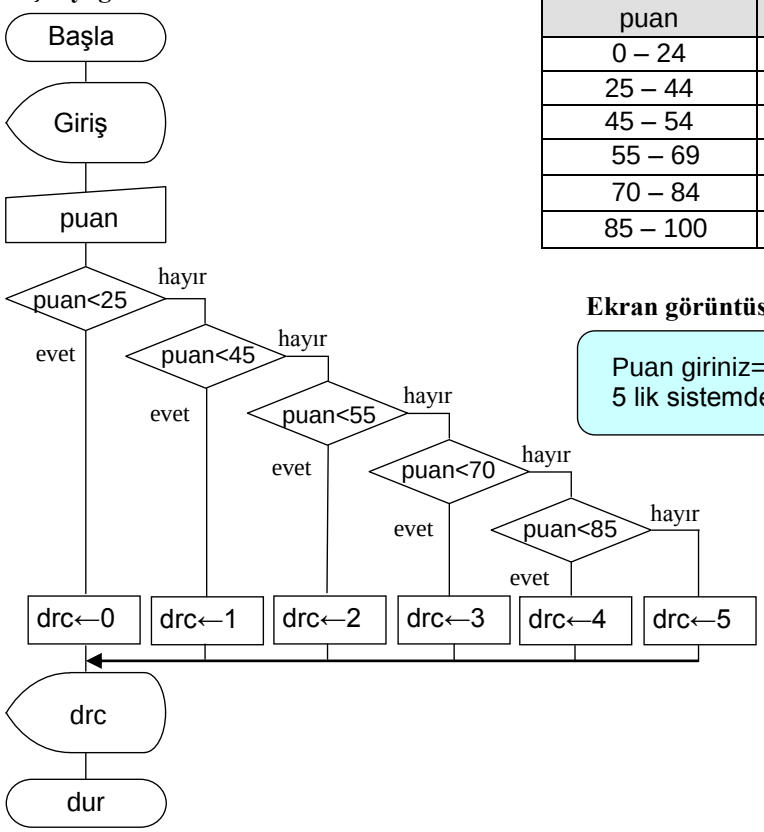
2.3. Çok yönlü karar yapısı

İki veya daha fazla durum için çok yönlü karar yapıları kullanılır. Bu yapıda, bir ifade diğer ifadenin durumuna bağlıdır.

Örnek:

Aşağıdaki tablodaki puan aralıklarına göre klavyeden girilen notun 5'lik sistemdeki karşılığını veren program yazınız.

Akış diyagramı



puan	derece
0 – 24	0
25 – 44	1
45 – 54	2
55 – 69	3
70 – 84	4
85 – 100	5

Ekran görüntüsü

```
Puan giriniz==>48
5 lik sistemdeki karşılığı 2
```

Program

```
/* örnek 2.3 */
#include<stdio.h>

main()
{
    int puan,drc;
    printf("Puan giriniz==>");
    scanf("%d",&puan);

    if (puan<25)                          //--1
       drc = 0;
       else if (puan<45)                  //--2
          drc = 1;
          else if (puan<55)               //--3
             drc = 2;
             else if (puan<70)            //--4
             drc = 3;
                 else if (puan<85)        //--5
                      drc = 4;
                 else                     //--6
                 drc = 5;
printf("5'lik sistemdeki karsiligi %d.\n",drc);
}
```

Program Açıklaması

Puanın girilmesinden sonra bu program, puan <25 şartını sağlayıp sağlamadığını kontrol eder. Bu karşılaştırma, doğru sonucu veriyorsa program bunu birinci değişken olarak atar; yanlış ise puan, ikinci if koşulu ile tekrar kontrol edilir. Bu durum son if koşuluna kadar devam eder. Son durum da yanlış sonuç verirse, son if koşulundan sonraki else icra edilir. Bu else veya else bloğu, gerekli değilse ihmale edilebilir. İç içe if yapısı ile çoklu seçim yönteminin kullandığı yapı konusunda dikkatli olunması gerekir. Çoklu yönlü karar yönteminin kod yazma aşamasında karar ifadelerinin aynı hizada değil de belirli bir boşluk kadar içeriden başlaması, programı daha anlaşılır hâle getirir.

2.4. Çoklu Karar Yapısı

Çok yönlü karar yapısındaki durumların kontrolü birbirine bağlı şartlara bağlıydı. Çoklu karar yapısında (switch-case) ise şartların hiçbir önceliği yoktur.

Örnek:

Aşağıda yer aldığı şekilde notları derecelendiren programı yazınız. Gerekli olan tablo aşağıda verilmiştir.

puan	derece
8-10	1
7	2
6	3
0 – 5	4

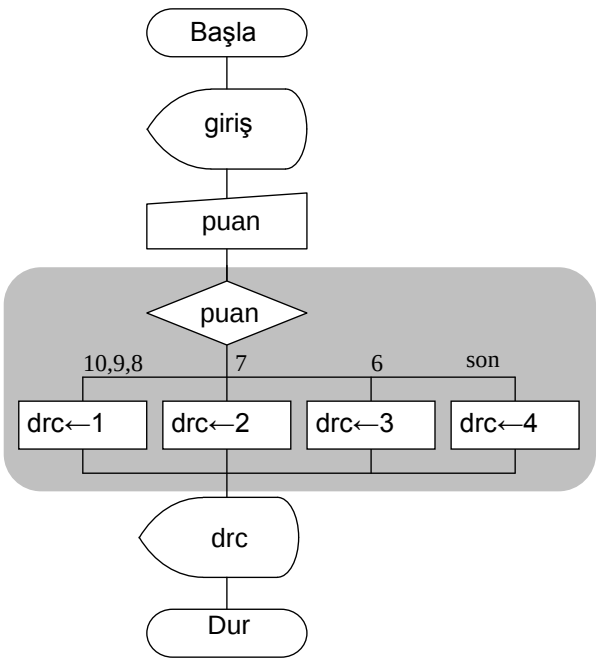
Ekran görüntüsü

```
Puan giriniz==>7
Dereceniz  2.
```

Değişken Tablosu

Değişken	Tür	Kullanım
puan	int	Not
drc	int	Nota karşılık gelen derece

Akış diyagramı



Program

```
/* Örnek 2.4*/

#include<stdio.h>
main()
{
     int not1,drc;
    printf("Puan giriniz==>");
    scanf("%d",&puan);
    switch (puan) {
        case 10:
        case 9:
        case 8:
            drc=1;
            break;
        case 7:
            drc=2;
            break;
        case 6:
            drc=3;
            break;
        default:
            drc=4;
    }
    printf("Dereceniz %d. \n",drc);
}
```

Program açıklaması

"Puan" değişkeninin değerine bağlı olarak yapılan çok yönlü karar yapısı akış diyagramında gösterilmiştir. "puan" değişkeninin değeri 10,9 veya 8 olduğunda drc←1 işlemi gerçekleştirilecektir ve 7 değeri için drc ←2 değeri aktarılacaktır. Benzer şekilde, 6 için drc←2 olacaktır. Son olarak puan değeri, son değerini(default) alacak ve bu şekilde işletilecektir.

Tüm bunlar program örneğinde C dilindeki switch-case yapısı içinde kodlanmıştır. Bu yapıda, "puan" değişkeni "switch" ifadesinden sonra parantez içerisine alınır ve "case" komutundan sonra kontrol edilecek değişkenin değerleri verilir. Bu değerlerden sonra ":" karakteri kullanılmak zorundadır.

Break komutu, komutların çalışmasını durdurarak "switch-case" yapısından dışarı çıkılmasına imkân verir. "Case" den sonra kullanılan tüm komutlar, break komutu ile karşılaşıncaya kadar işletilir. Bu programda, **case 10: case 9:** ve **case 8**: ifadelerinden sonra break komutu yoktur. Tüm bu değerler, aynı komutun icra edilmesini sağlar. (drc=1;)

Switch-case yapısında, case ifadesinden sonra integer bir değer gibi 'A' karakterini de yazabiliriz. A karakteri, tek tırnak arasında olmalıdır. Bu tür bir yazım, karar yapılarında karakter kodunun kontrol edilmesini sağlar. Karakter kodu hakkında ileride daha detaylı durulacaktır.

```
char  ch1;
ch1='A';
```

Bir karakteri saklayan değişken, bir **karakter tip** değişkenidir ve **char** tipi ile tanımlanır. scanf ve printf komutundaki ve tip dönüşüm belirteci karakter için %c'dir. Aşağıdaki program, bir malın kodunu kontrol ederek malın cinsini yazar.

```
#include <stdio.h>
main()
{
    char kod;
    printf("Kodu giriniz==>");
    scanf("%c",&kod);

    switch(kod){
    case 'A':
          printf("Kod %c --- TV\n",kod);
    break;
    case 'B':
        printf("Kod %c --- Radyo\n",kod);
    break;
default:
    printf("Kod %c --- diğer\n",kod);
   }
}
```

Ekran görüntüsü

```
Kodu giriniz==>B
Kod  B --- Radyo
```

UYGULAMA FAALİYETİ

Aşağıdaki sorulara ilişkin uygulama faaliyetini yapınız.

- Klavyeden girilen iki tam sayıdan büyük olanı ekrana yazan programı yazınız.

- İki sınavın notlarının klavyeden girilmesi istenmektedir. Öğrencinin başarılı olabilmesi için gereken şartlar şunlardır.

  1. Toplam sonuç 140 ve daha büyük
  2. İki not da 65'ten küçük olmayacak.

  Bu şartlara göre ekrana başarılı veya başarısız yazan programı yazınız.

- Aşağıdaki tablo, bir otobüs hattındaki uzaklıklara göre yol ücretini göstermektedir. Bu tabloya göre klavyeden uzaklığı km cinsinden girilen bir yere gitmek için ödeyeceğimiz yol parasını hesaplayan programı yazınız.

Uzaklık	Yol ücreti
0.0 - 9.9km	**2 YTL**
10.0 - 29.9km	**3 YTL**
30.0 - 49.9km	**4 YTL**
50km -	**5 YTL**

- Girilen gelirin büyüklüğüne göre aşağıda belirtilen oranlarda gelir vergisi alınacaktır. Klavyeden girilen gelire göre ödenmesi gereken gelir vergisini hesaplayan programı yazınız.

  Gelir vergisi için tanımlanan kurallar aşağıda verilmiştir.

Gelir 10,000 YTL den düşük ise vergi yok.
10,000 YTL- 299,999 YTL 5%
300,000 YTL- 499,999 YTL 8%

Bundan sonra her 200,000 YTL artışı vergi oranını 1% artıracaktır.

Örneğin 800,000 YTL için 10% olacaktır.
Not: Gelir vergisi maksimum 80 % olabilir.

İşlem Basamakları	Öneriler
➢ Değişken tablosunu hazırlayınız.	➢ Programda kullanacağınız değişkenlerin tipini belirleyiniz. ➢ Değişken isimlendirme kurallarına dikkat ediniz.
➢ Akış diyagramını çiziniz.	➢ Akış diyagramı sembollerinden yararlanınız.
➢ Programı yazınız.	➢ Program satırlarının düzenli olmasına özen gösteriniz. ➢ Matematiksel işlem önceliklerine dikkat ediniz. ➢ Karar ifadelerinin belirlenen şartlara uygun olmasına dikkat ediniz.
➢ Yazdığınız programı derleyiniz.	
➢ Programda hata var ise bunları gideriniz.	
➢ Ekran görüntüsünü kontrol ediniz.	

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri **Evet**, kazanamadığınız becerileri **Hayır** kutucuğuna (**X**) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Karar yapılarını uygulayabildiniz mi?		
2. Çoklu karar uygulamasını yapabildiniz mi?		
3. Birden çok karar uygulamasını yapabildiniz mi?		
4. Matematiksel işlemlerde işlem önceliğini yapabildiniz mi?		
5. Program yazarken akış diyagramından yararlandınız mı?		
6. Değişken isimlendirme yapabildiniz mi?		
7. İstenilen sonucu ekran çıktısını alabildiniz mi?		
8. Teknolojik kurallara uygun bir çalışma gerçekleştirdiniz mi?		
9. Süreyi iyi kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda "Hayır" şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız "Evet" ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Bu faaliyet sonunda kazandıklarınızı aşağıdaki soruları cevaplandırarak ölçünüz.

1. Aşağıdaki şart ifadelerine karşılık gelen if komutlarını eşleştirme yaparak bulunuz.
A) a değişkeninin değeri b değişkeninin değerine eşit olduğunda
B) a değişkeninin değeri 15 veya yukarısı olduğunda
C) a değişkenin değeri 15'ten küçük olduğunda
D) a değişkeninin değeri 15 değerinden büyük olduğunda
E) a değişkeninin değeri 15'ten farklı olduğunda

A) if(a>15) **B)** if(a<15) **C)** if(a=b) **D)** if(a= =b)

E) if(a>=15) **F)** if(a!=15) **G)** if(a=!15) **H)** if(a=>15)

2. **Sayi** değişkeninin 10'a eşit olduğu ve **toplam** değişkeninin 20'den küçük olduğu durumlarda **"yanlıs."**, kelimesini basan ifadeyi tamamlayınız.

```
if ((sayi ___10) _____ (toplam ___20))
                printf("yanlıs.");
```

3. **not1** değişkeninin 80'den fazla veya **not2** değişkeninin 60'tan fazla olduğu durumlarda **"dogru"** kelimesini basan ifadeyi tamamlayınız.

```
if ((not1 ___80) _____ (toplam ___60))
    printf("dogru ");
```

4. **not1** değişkeninin 20'den küçük olmadığı ve 40'tan büyük olmadığı durumda "geçerli sayı " kelimelerini basan ifadeyi tamamlayınız.

```
if ((not1 ___20) _____(not1 ___40))
    printf("gecerli sayi");
```

5. Aşağıdaki ifadeyi switch komutunu kullanarak tamamlayınız.

```
if( urun == 'A' )
      fiyat = 200;
else if ( urun == 'B' )
      fiyat = 50;
else if( urun == 'C' )
      fiyat = 600;
else
printf("Bilinmeyen urun -->%c\n", urun );
```

```
switch( __________ )
{
     case ____:  fiyat =200; break;
     case 'B'    : fiyat= 50; break;
     case 'C'    : fiyat =600; break;
    ________ printf( "Bilinmeyen urun -->%c\n", urun ); break;
}
```

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-3

AMAÇ

Bilgisayar programlamada döngü yapılarını doğru bir şekilde kullanabileceksiniz.

ARAŞTIRMA

- C programlama dilinde döngü yapıları hakkında araştırma yapınız.

3. DÖNGÜLER

Döngü (loop), programlamada en önemli yapılardan birisidir. Bazen birçok verinin çıktısının alınmasında veya kalem kâğıtla yapamayacağımız birçok işlemin yapılmasını sağlayan araçtır. Döngü yapısını tasarlamak, program yaparken karşılaşacağımız en zor ve en önemli durumlardan birisidir. Bu yüzden çok miktarda pratik yapmaya ihtiyacınız vardır.

3.1. While Döngüsü

Örnek:

"Merhaba!" kelimesini ekrana 5 kez yazan programı yapınız.

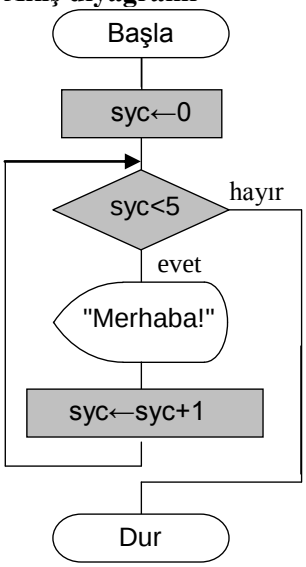
Değişken tablosu

Değişken	Tür	Kullanım
syc	int	Sayaç

Ekran görüntüsü

```
Merhaba!
Merhaba!
Merhaba!
Merhaba!
Merhaba!
```

Akış diyagramı

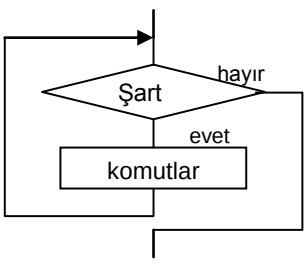


Program

```
/* Örnek 3.1 */

#include <stdio.h>

main()
{
    int syc;

    syc=0;
    while(syc < 5)
    {
        printf("Merhaba!|n");
        syc = syc + 1;
    }
}
```

Program Açıklaması

Genel olarak döngü oluşturmak için kullanılan komutlardan birisi while yapısıdır. Akış diyagramında görüldüğü gibi şart ifadesi döngünün içerisine girmeden önce kontrol edilmektedir. Şart ifadesi doğru olduğunda, döngü içerisindeki komutlar şart ifadesi yanlış oluncaya kadar icra edilecekleridir. Diğer durumda içerdeki komutlar işletilmeyecektir. Bu duruma "işletilen durum" adı verilir. Bu komutun yapısı, if yapısı ile aynı şekildedir. Aşağıdaki tabloya göre akış diyagramındaki while komut yapısı kolaylıkla koda dönüştürülebilir.

Akış Diyagramı	C dilindeki karşılığı
	**while(Şart ifadesi)** **{** **Komutlar** }

Şekil 3.1: While yapısı

İçinde döngü kullanılması gereken bir program tasarlanırken ilk olarak döngünün hangi değerden başlayacağını, döngünün nerede sonlandırılacağını ve döngünün nasıl tekrar edileceğini düşünmeliyiz. Programı yukarıdan aşağıya doğru satır satır yazmaya çalıştığınızda istediğiniz sonuca elde etmeniz mümkün değildir. Program yapmak bulmaca

çözmek gibidir. Bu yüzden öncelikle koyması veya düşünülmesi kolay olan parçaları tamamlamamız gerekir. Bu şekilde bir çalışmayı alışkanlık haline getirdiğinizde yaptığınız programlarda mantıksal olarak hata yapmazsınız. Programımızda önemli değişkenleri tanımladıktan sonra döngü yapısını oluşturmalısınız.

Örnek programda, döngü yapısını incelemek istersek syc değişkeni üç kere kullanılmıştır.

1. syc=0 ---- döngüden önce
2. while(syc < 5) ---- while komutunda
3. syc = syc + 1 ---- döngünün son satırında

syc = 0 değeri döngüden önce sadece bir kez işletilir. Bu durumda değişkene 0 değeri (başlangıç değeri) atanmıştır.

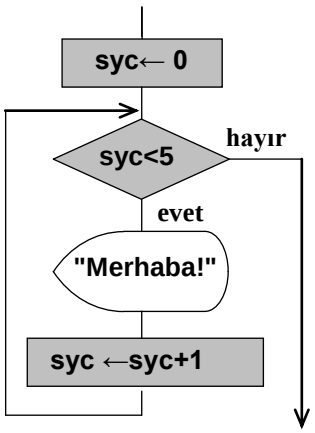
while(syc < 5) Şartın kontrol edildiği durum

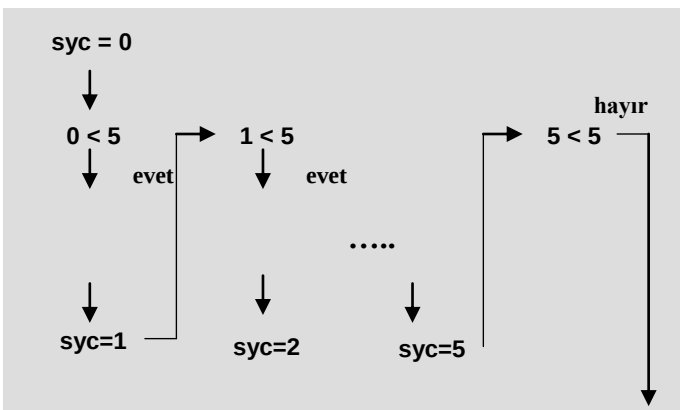
syc = syc + 1 Matematikte imkânsız olan bu ifade, bilgisayar tarafından syc değerine sürekli 1 sayısının eklenmesi anlamındadır.

Döngü yapısı için bu üç ifade çok önemlidir.

Programın İzlenmesi

Aşağıdaki akış diyagramını incelerseniz tekrarın 5 kez yapıldığını görebilirsiniz.





Yukarıda adım adım yapılan yönteme izleme metodu (tracing) adı verilir. Adım adım izlemede kendinizi bilgisayarın yerine koyabilirsiniz. Program yazımından önce akış diyagramı çizerek bu yöntemi kullanmak, döngü yapısını öğrenmeniz açısından önemlidir.

Programlarda sayaçlar çok sık kullanılır. C dilindeki basit sayaç kullanımı aşağıda görülmektedir.

```
syc++
```

Burada artırma operatörü kullanılmıştır. Aynı şekilde azaltma operatörü de vardır.

Operatör	Kullanım	Anlam
++ (Arttırma)	i++	i = i + 1
-- (Azaltma)	i --	i = i - 1

Şekil 3.2: Arttırma ve azaltma operatörü

Örnek:

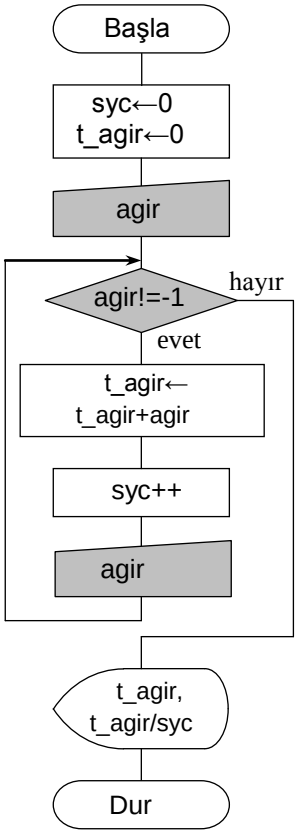
Girilen parça ağırlıklarına göre toplama ağırlık ve ortalama ağırlık hesaplamalarını yapan programı yazınız. -1 değeri girildiğinde bilgi girişi duracaktır.

Değişken	Tür	Kullanım
agir	float	Parça Ağırlığı
t_agir	float	Parçaların toplam ağırlığı
syc	int	Sayaç

Ekran görüntüsü

```
Parça ağırlığı(bitir- -1)=>3.2
Parça ağırlığı(bitir- -1)=>3.3
Parça ağırlığı(bitir-- 1)=>3.4
```

Akış diyagramı



Program

```
/* Örnek 3.2 */

#include <stdio.h>

main()
{
      int syc=0;
      float agir, t_agir=0;
      printf("parca agirligi (bitir----1)=>");
      scanf("%f",&agir);
      while(agir !=-1)
      {
          t_agir = t_agir + agir;
          syc++;
          printf("parça agirliği (bitir----1)=>");
          scanf("%f",&agir);
       }
      printf("Toplam agirlik   =%6.1f\n",t_agir);
      printf("Ortalama agirlik =%6.1f",t_agir/syc);
}
```

Program Açıklaması

1) Döngü Yapısı

Bu döngüyü tasarlarken anahtar değişken, ağırlık değerini kullanacak agir değişkenidir. Bu durumda while döngüsünün şart ifadesi aşağıdaki şekilde yazılabilir.

agir != -1.

Akış diyagramında ağırlık girişi iki defa görülmektedir. Bir önceki programda olduğu gibi başlangıç değeri ve diğeri de artışın yapıldığı durumdur. Böylece bir önceki örnek ve bu örneği aynı yapıya sahip diye tanımlayabiliriz. Bu durumda iki defa değer girişi çok hoş durmamaktadır. Bu yapı bize programın kolayca yapılmasını sağlayacaktır.

2) Toplam Ağırlığın Bulunması

Toplam ağırlığın bulunması için iki hesaplamanın yapılması zorunludur.

t_agir = 0 (başlangıç değeri)
t_agir = t_agir + agir

İkinci hesaplama için yeni ağırlık değeri toplam ağırlığa eklenecek bir sonraki değeri ise yine bu toplam değere ilave edilerek sürekli bir toplama işlemi yapılacaktır.

3) Sayaç

Sayaç işlemi bir önceki programla aynı işlemi yapmaktadır.Bu durumda akış diyagramını tekrar incelemeniz ve döngünün girilen ağırlık sayısını sayıp saymadığını kontrol etmeniz gerekir.

4) Yazma

Toplam ağırlık ve sayaç kullanarak bizden istenilen toplam ağırlık ve ortalama ağırlık değerleri yazdırılabilir.

Örnek:

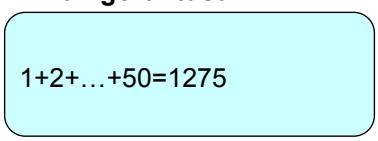
Aşağıdaki işlemin sonucunu bulan programı yazınız.

1+2+3……+50

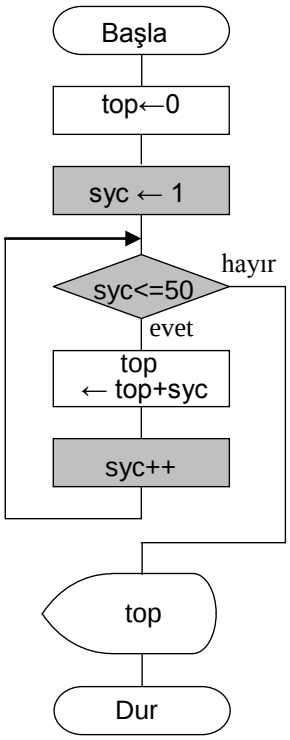
Değişken tablosu

Değişken	Tür	Kullanım
syc	int	Eklenen sayı
top	int	Toplam sonuç

Ekran görüntüsü



Akış diyagramı



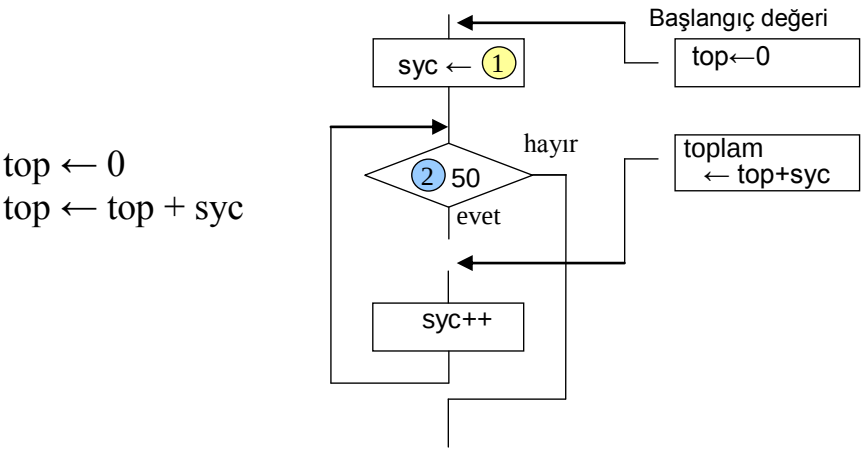
Program

```
/* Örnek 3.3 */

#include <stdio.h>

main()
{
     int top=0,syc;

     syc=1;
     while (syc<=50)
     {
          top=top+syc;
          syc++;
      }
     printf("1+2+...+50=%d\n", top);
}
```

Program Açıklaması

Bir döngü için kullanılan değişkene kontrol değişkeni adı verilir. Bu programda syc değişkeni, sayıları üretmek ve bunları kontrol değişkenine eklemek için kullanılmaktadır. Tasarım için öncelikle döngü işlemi şekil 3.3'de görüldüğü gibi hazırlanır. Tabloda 1 ve 2 bölümleri boş bırakılmıştır.

Daha önce de çalıştığımız gibi toplamın bulunması ve iki elemana ihtiyacımız vardır.



Şekil 3.3: Artırma ve azaltma operatörü

Bu değişkenlerden birincisinin döngüden önce, diğerinin ise döngünün içerisinde olması düşünülmüştür. Son olarak 1 ve 2 numaralı bölümler son syc değeri için düşünülmelidir. 1'den 50'ye kadar olan sayıların toplamı bulunurken (50 dâhil) dikkat edilmesi gereken başlangıç değeri ve bitiş değerleridir. (Başlangıç değeri) olabilir ve 2 numaralı bölüm <= ile doldurulmalıdır. Son aşama için akış diyagramını dikkatlice takip etmelisiniz.

3.2. For Döngüsü

Örnek:

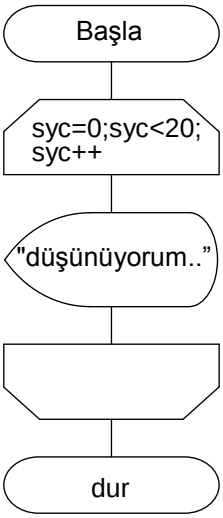
Düşünüyorum kelimesini alt alta 20 defa ekrana basan programı yazınız.

Değişken tablosu		
Değişken	**Tür**	**Kullanım**
syc	int	Sayma işlemi(kontrol değişkeni)

Ekran görüntüsü

```
düşünüyorum...
düşünüyorum...
düşünüyorum...
düşünüyorum...
düşünüyorum...
```

Akış diyagramı

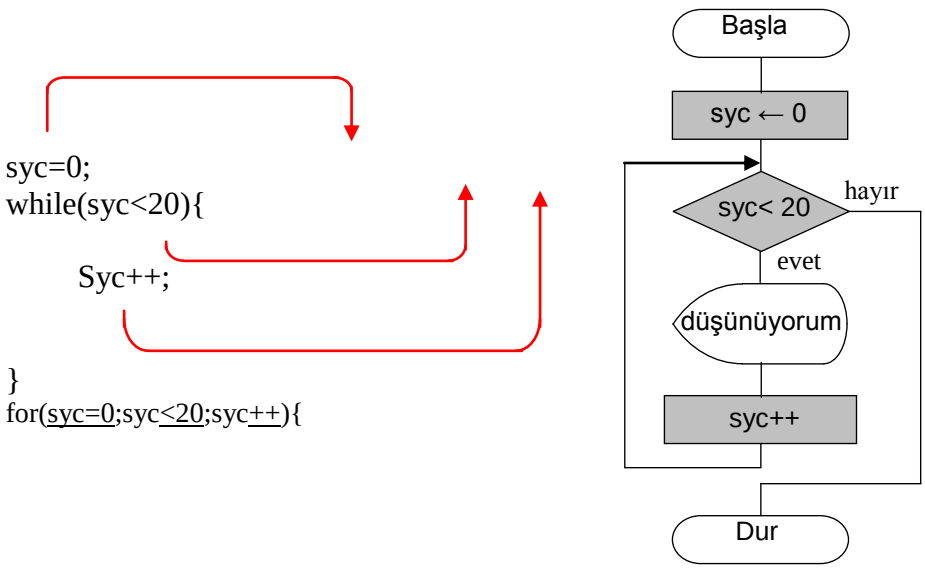


Program

```
/* Örnek: */

#include<stdio.h>

main()
{
    int syc;

    for(syc = 0; syc < 20;syc++)
    {
        printf("dusunuyorum…\n");
     }
}
```

Program Açıklaması

Bu örnek while döngüsünün yerine for döngüsü ile yapılan temel sayma programıdır. Bunun için daha önceden tanıştığımız gerekli olan üç eleman vardır. Bunlar while yapısını kullanan aşağıdaki örnekle ilişkilidir

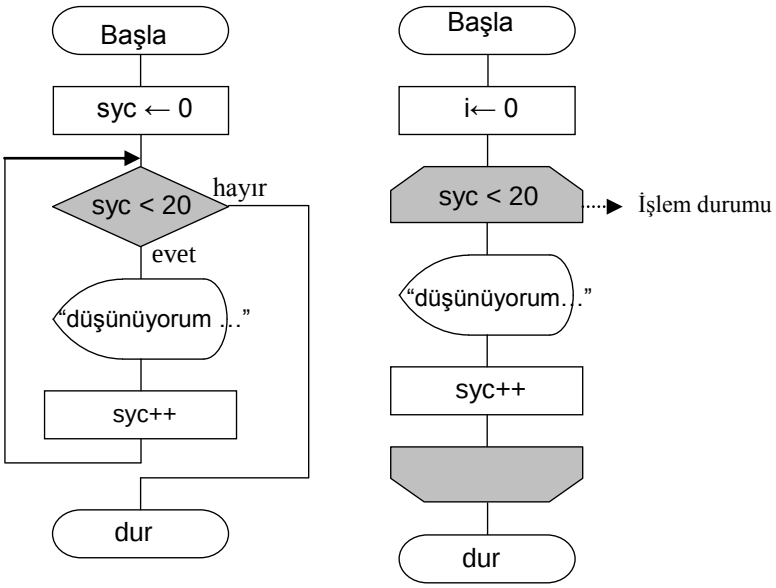


Şekil 3.1: While ve for döngü yapıları

Aşağıdaki örnekte görüldüğü gibi for yapısı içinde sadece bir komut varsa süslü parantezler kullanmayabiliriz.

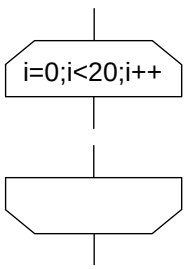
```
for(i = 0; i < 20;i++)
    printf("düşünüyorum …\n ");
```

Şimdiye kadar kullandığımız döngü akış diyagramlarını çizmek zaman almaktadır ve çizimi takip açısından biraz zordur. Bu yüzden bundan sonra sağdaki akış diyagramını döngüleri tanımlamak için kullanacağız.



Şekil 3.2: Döngü için akış diyagramları

Bu tür bir kullanım döngü yapısını açıkça gösterdiği ve bizi önemli elemanları yazmayı unutmamızı engellediği için tercih edilen bir yöntemdir. Akış diyagramında, **for** döngüsü için gerekli şartlar sağda gösterildiği gibi yerleştirilmiştir.



For döngüsü için akış diyagramı

Örnek:

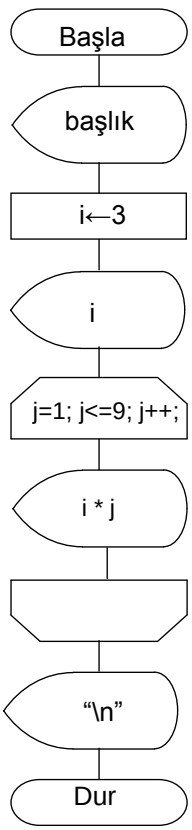
For döngüsü ile yandaki şekildeki gibi 3 değeri için çarpım tablosu oluşturan programı yazınız.

Ekran görüntüsü

```
      1  2  3  4  5  6  7  8  9
3     3  6  9  12 15 18 21 24 27
```

Değişken tablosu		
Değişken	**Tür**	**Kullanım**
i	int	3 değeri için
j	int	1 – 9 sayılarını üretmek için (Kontrol Değişkeni)

Akış diyagramı



Program

```
/* Örnek */

#include<stdio.h>

main()
{
    int i,j;
    printf("  1  2  3  4  5  6  7  8  9\n");        //--1
    i=3;                                            //--2
    printf("%d",i);                                 //--3
    for(j = 1; j<= 9;j++)                           //--4
        printf("%3d",i*j);
    printf("\n");                                   //--5
}
```

Program Açıklaması

1. İlk satırın ekranda gösterilmesi.

2. i değişkeni için 3 değerinin atanması.

3. \n elemanını kullanmadan i değişkeninin görüntülenmesi.

4. Değişen j değerine göre for döngüsü içinde (i × j) değerinin hesaplatılması

5. "\n" ile satırın değiştirilmesi.

3.2.1. İç İçe Döngüler

Örnek:

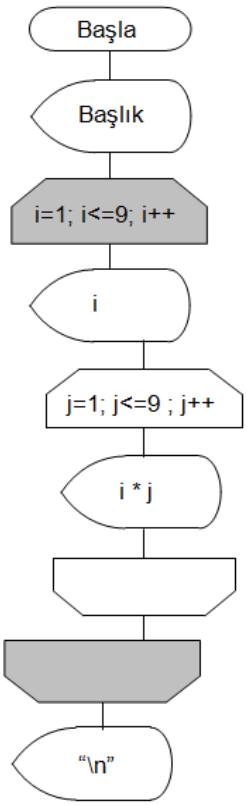
Ekran çıktısında görülen çarpım tablosunun programını yazınız.

Ekran görüntüsü

```
    1  2  3  4  5  6  7  8  9
1   1  2  3  4  5  6  7  8  9
2   2  4  6  8  10 12 14 16 18
3   3  6  9  12 15 18 21 24 27
       :
9   9  18 27 36 45 54 63 72 81
```

Değişken tablosu		
Değişken	**Tür**	**Kullanım**
İ	int	1 – 9 arasındaki sayılar için (Dış döngünün kontrol değişkeni)
j	int	1 – 9 arasındaki sayıları için (İç döngünün kontrol değişkeni)

Akış diyagramı



Program

```
/* Örnek*/
#include <stdio.h>
main()
{
        int i,j;

        printf("  1  2  3  4  5  6  7  8  9\n");

        for(i = 1; i <= 9;i++)
        {
                printf("%d",i);
                for(j = 1; j <= 9; j++)
                        printf("%3d", i*j);
                printf("\n");


        }
}
```

Program Açıklaması

1 ile 9 arasındaki sayıları üretip kolon olarak kullanmak için dış döngüyü kullanabiliriz.

Örnek:

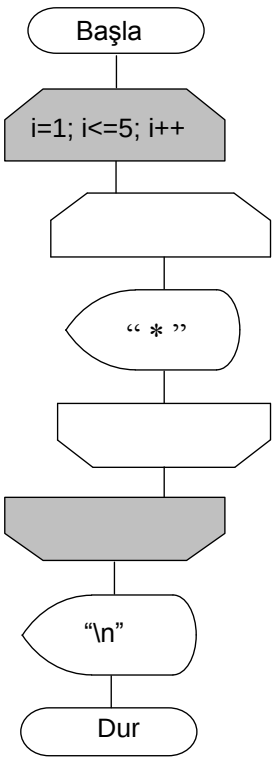
Yandaki şekilde görüldüğü gibi “ * ” karakterini ekrana basan programı yazınız.

Ekran görüntüsü

```
*
**
***
****
*****
```

Değişken tablosu		
Değişken	**Tür**	**Kullanım**
İ	int	Satır için kullanılan değişken
j	int	Sütun için kullanılan değişken

Akış diyagramı



Program

```
/* Örnek */

#include <stdio.h>

main()
{
    int i,j;

    for(i = 1; i <= 5;i++)
    {
        for (j = 1; j <=i; j++)
          printf("*");
        printf("\n");
     }
}
```

Program Açıklaması

Şekildeki yapıyı “*” karakteri ile oluşturmak için iki döngüye ihtiyaç vardır. Programda i değişkeni satırlar için j değişkeni ise sütunlar için kullanılır. Dış döngüdeki i değişkeni 1'den 5 değerine kadar artan şekildedir. Döngüde i=1 olduğunda iç döngü j=1 değerinden başlayacak ve j=i➔1 olduğu için sadece 1 adet “*” karakteri yazacaktır.

Program 2. satıra printf("\n”) komutu ile geçer. Bu durumda dıştaki döngüde i=2 olmuştur. İç döngüde ise j=i➔2 olduğu için 2 defa “*” karakterine basar.

Bu işlem dış döngüde i=4 değerine kadar tekrarlanır.

Örnek:

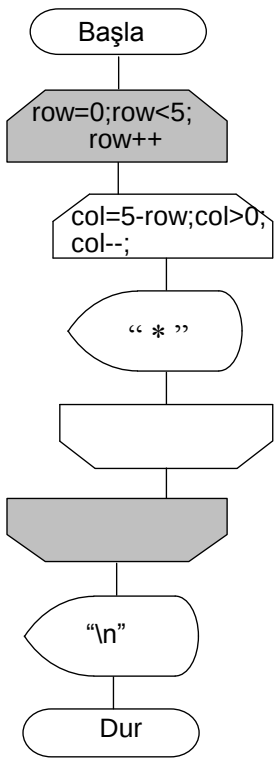
Yandaki şekilde görüldüğü gibi “ * ” karakterini ekrana basan programı yazınız.

Ekran görüntüsü

```
*****
****
***
**
*
```

Değişken tablosu		
Değişken	**Tür**	**Kullanım**
row	int	Satır için kullanılan değişken
col	int	Sütun için kullanılan değişken

Akış diyagramı



Program

```
/* Örnek 3.8 */

#include<stdio.h>

main()
{
        int row,col;

        for(row=0;row<5;row++)
        {
                for(col=5-row;col>0;col--)
                        printf("*");
        printf("\n");
         }
}
```

Program Açıklaması

Bu örnek yapı olarak önceki örneğin benzeriydi. Program dıştaki döngüden başlar. Dış döngüdeki **row** değişkeni **0** değerinden başladığı için içteki döngüde **col** değişkeni **col** =**5-0** ➔5'den 1'e kadar 5 defa ***** karakterini yazar. Program 2. satıra **printf("\n")** komutu ile geçer. Bu durumda dıştaki döngüde **row=1** olmuştur. İç döngüde ise **col**=5-1➔4 değerinden 1 değerine kadar 4 defa **“*”** karakterine basar.

Bu işlem dış döngüde **row=4** değerine kadar tekrarlanır.

UYGULAMA FAALİYETİ

Aşağıdaki sorulara ilişkin uygulama faaliyetini yapınız.

- Birim fiyatları ve satış rakamları girilen malların toplam tutarını hesaplayan programı yazınız(klavyeden girilen verilerin sonlandırma işlemi 0 ile yapılabilir).

```
Birim fiyat          ==>2000
Satış adedi          ==>12
Birim fiyatı         ==>1500
Satış adedi          ==>8
   :
Birim fiyat          ==>0(veri sonu)
Toplam tutar  ==>????
```

- 1 ile 100 arasındaki tek sayıları toplayan programı yazınız.

- Aşağıdaki işlemin sonucunu bulan programı yazınız.

  1x2x3x4x5x…..x50

- Aşağıda görüldüğü gibi toplam 10000 YTL elde edebilmemiz için gereken gün sayısını hesaplayınız. İlk gün 1 YTL kazandığımız bir sonraki günde bu değerin iki katı kazandığımız varsayılıyor.

Kazanç		**Toplam kazanç**
1. gün	1	1
2. gün	2(= 2 × 1)	3 (= 1 + 2)
3. gün	4(= 2 × 2)	8 (= 3 + 4)

- Ekrana aşağıdaki şekilde olduğu gibi * karakterini basan programı yazınız.

```
    *
   **
  ***
 ****
*****
```

```
    *
   ***
  *****
 *******
*********
```

İşlem Basamakları	Öneriler
➢ Değişken tablosunu hazırlayınız.	➢ Programda kullanacağınız değişkenlerin tipini belirleyiniz. ➢ Değişken isimlendirme kurallarına dikkat ediniz.
➢ Akış diyagramını çiziniz.	➢ Akış diyagramı sembollerinden yararlanınız.
➢ Programı yazınız.	➢ Program satırlarının düzenli olmasına özen gösteriniz. ➢ Matematiksel işlem önceliklerine dikkat ediniz. ➢ Karar ifadelerinin belirlenen şartlara uygun olmasına dikkat ediniz. ➢ Döngü başlangıç, sınır ve artış değerlerine dikkat ediniz.
➢ Yazdığınız programı derleyiniz.	
➢ Programda hata var ise bunları gideriniz.	
➢ Ekran görüntüsünü kontrol ediniz.	

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdaki for ifadelerinden hangisi alt alta 1' den 10 ' a kadar olan sayıları ekrana yazdırmaya yarar?

A)
```
for( say = 1; say <= 10; say ++)
    printf("%d\n", say);
```

B)
```
for( say = 1; say < 10; say ++)
    printf("%d\n", say);
```

C)
```
for( say = 0; say <= 9; say ++)
    printf("%d ", say);
```

D)
```
for( say = 1; say <> 10; say ++)
    printf("%d\n", say);
```

2. Aşağıdaki çıktıyı verecek olan ifade aşağıdakilerden hangisidir? (İp ucu: iç-içe döngüler kullanılacak)

```
1
22
333
4444
55555
```

A)
```
for(a = 1; a <= 5; a = a + 1)
  {
    for( b = 1; b <= 5; b = b + 1)
      printf("%d", b);
    printf("\n");
  }
```

B)
```
for( a = 1; a <= 5; a = a + 1)
  {
    for( b = 1; b <= a; b = b + 1)
      printf("%d", a);
    printf("\n");
  }
```

C)
```
for( a = 1; a <= 5; a = a + 1)
  {
    for( b = a; b <= 5; b = b + 1)
      printf("%d", b);
    printf("\n");
  }
```

D)
```
for( a = 1; a <= 5; a = a + 1)
  {
    for( b = 1; b < a; b = b + a)
      printf("%d", b);
    printf("\n");
  }
```

3. 10 ile 100 arasındaki sayıların toplamını bularak, bu değeri toplam değişkenine atayan ifade aşağıdakilerden hangisidir? *toplam* değerinin başlangıç değeri program içerisinde verilecektir!.

A) for(a = 10; a <= 100; a = a + 1)
toplam = toplam + a;

B) for(a = 10; a < 100; a = a + 1, toplam = 0)
toplam = toplam + a;

C) for(a = 10; a <= 100, toplam = 0; a = a + 1)
toplam = toplam + a;

D) for(a = 10, toplam = 0; a <= 100; a = a + 1)
toplam = toplam + a;

4. Aşağıdaki ifadelerden hangisi A-Z arasındaki karakterleri ekrana yazmaya yarar?,

A) for(a = 'A'; a < 'Z'; a = a + 1)
printf("%c", a);

B) for(a = 'a'; a <= 'z'; a = a + 1)
printf("%c", &a);

C) for(a = 'A'; a <= 'Z'; a = a + 1)
printf("%c", a);

D) for(a = 'Z'; a <= 'A'; a = a + 1)
printf("%c", a);

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise "Modül Değerlendirme"ye geçiniz.

MODÜL DEĞERLENDİRME

UYGULAMALI TEST

Modülde yaptığınız uygulamaları tekrar yapınız. Yaptığınız bu uygulamaları aşağıdaki tabloya göre değerlendiriniz.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri **Evet**, kazanamadığınız becerileri **Hayır** kutucuğuna (**X**) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. include komutu ile istenilen kütüphaneleri belirttiniz mi?		
2. Main fonksiyonu içerisine ana programı yazdınız mı?		
3. Matematiksel işlem yaparken operatörleri kullandınız mı?		
4. printf() fonksiyonu ile verilerin ekranda çıktısını aldınız mı?		
5. scanf() fonksiyonu ile klavyeden veri okuttunuz mu?		
6. İki yollu karar yapısını kullandınız mı?		
7. İç içe karar yapısını kullandınız mı?		
8. Çoklu karar yapısını kullandınız mı?		
9. While döngü yapısını kullandınız mı?		
10. For döngü yapısını kullandınız mı?		
11. İç içe döngü yapısını kullandınız mı?		
12. Program yazım işlemlerinin doğruluğunu kontrol ettiniz mi?		

DEĞERLENDİRME

Değerlendirme sonunda "Hayır" şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız "Evet" ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ 1'İN CEVAP ANAHTARI

1	**C**
2	**A**
3	**C**
4	**A**
5	**D**
6	**B**
7	**B**
8	**A**
9	**C**
10	**C**

ÖĞRENME FAALİYETİ 2'NİN CEVAP ANAHTARI

1	**1-D**		
	2-E		
	3-B		
	4-A		
	5-G		
2	==	**&&**	<
3	<	||	<
4	>=	**&&**	<=
5	**mektup**	**'X'**	**default**

ÖĞRENME FAALİYETİ 3'ÜN CEVAP ANAHTARI

1	**A**
2	**B**
3	**D**
4	**C**

KAYNAKÇA

- ISHIDA Yasuhiro, Hideki MURAKAMI, Ito KOICHI, Gürcan BILDIR, **Bilgisayar Kontrol Teknolojisi,** M.E.B – JICA, 2005.