T.C.
MİLLÎ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

AKIŞ DİYAGRAMLARI

ANKARA 2007

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).

- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.

- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.

- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşalabilirler.

- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.

- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**481BB0025**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Alan Ortak**
MODÜLÜN ADI	**Akış Diyagramları**
MODÜLÜN TANIMI	Programlama akış şeması ile ilgili öğrenme materyalidir.
SÜRE	40/24
ÖN KOŞUL	Programlama Temelleri modülünü almış olmak.
YETERLİK	Akış diyagramlarını kullanmak
MODÜLÜN AMACI	**Genel Amaç** Gerekli ortam sağlandığında, veri giriş çıkış işlemleriyle, bilgisayarda ki komut ve değişkenleri tanıyıp, basit programlar yapabileceksiniz. **Amaçlar** 1. Veri giriş ve çıkış işlemleri yapabileceksiniz. 2. Değişkenler, sabitler ve açıklama satırlarını kullanabileceksiniz. 3. Sayılar ve metin veri türü ile çalışabileceksiniz. 4. Kontrol deyimleri ile program akışını değiştirebileceksiniz. 5. Döngüleri kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Bilgisayar laboratuvarı ve bu ortamda bulunan, bilgisayar, yazıcı, bilgisayar masaları, kâğıt, kalem, lisanslı işletim sistemi programı ve akış diyagramı sembolleri ile ilgili panolar.
ÖLÇME VE DEĞERLENDİRME	Her faaliyet sonrasında o faaliyetle ilgili değerlendirme soruları ile kendi kendinizi değerlendireceksiniz. Modül içinde ve sonunda verilen öğretici sorularla edindiğiniz bilgileri pekiştirecek, uygulama örneklerini ve testleri gerekli süre içinde tamamlayarak etkili öğrenmeyi gerçekleştireceksiniz. Sırasıyla araştırma yaparak, grup çalışmalarına katılarak ve en son aşamada alan öğretmenlerine danışarak ölçme ve değerlendirme uygulamalarını gerçekleştireceksiniz.

GİRİŞ

Sevgili Öğrenci,

Her şeyden önce herkes bir programlama dilini öğrenebilir. Düşünüldüğünün aksine bilgisayar programcılığı yüksek bir zekâ ve matematik bilgisi gerektirmez. Önemli olan kişide öğrenme isteği olmasıdır ve öğrenen kişinin hemen vazgeçmemesidir.

Programlama bir hünerdir. Bazı insanlar doğal olarak diğerlerinden daha iyidir, ama herkes pratik yaparak iyi olabilir. Başaramamaktan korkmayınız, kendinizi bu maharete vererek, öğrenmek için çaba göstermeniz yeterli olacaktır. Programlama eğlencelidir, fakat basit hatalardan dolayı sinir bozucu olabildiği gibi zamanını da boşa geçmesine neden olabilir. Bu sebeple bu modülleri takip ederek, en az sıkıntı ve en yüksek memnuniyet ile programlamayı öğrenebilirsiniz.

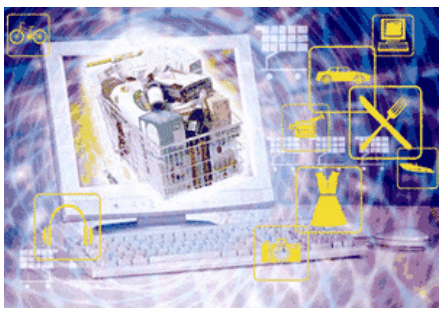
Bu modül ile kazanacağınız konular giriş çıkış işlemleri, değişkenler, işleçler, fonksiyonlar, akış kontrol deyimleri ve döngülerdir. Modülü bitirdiğinizde anlamadığınız yerleri tekrar okuyup, uygulayınız. Konuları tam olarak kavramadan diğer modüle geçmeyiniz.

Konular kapsamlı olarak, derinlemesine anlatılmamıştır. Ne kadar çok uygulama, araştırma yapar iseniz kendinizi o kadar geliştirirsiniz ve ilerletirsiniz.

Her dilin kendine göre avantajı bulunmaktadır. Modülde verilen örnekleri yaptığınızda o dilleri öğrenmiş olmayacaksınız. Asıl istenen, anlatılmak istenen konunun uygulanması somut ve anlaşılır hâle gelmesidir.

Bu modülde sağlık, muhasebe, internet ve matematik ile ilgili örnekler verilmektedir. Programlarda bilerek veya bilmeyerek yapılmış hatalar olabilir. Bulduğunuz hataları ve yeni önerilerinizi arkadaşlarınızla paylaşınız.

Modül içinde geçen araştırma konuları için “Önerilen Kaynaklar” kısmından yardım almayı unutmayınız.

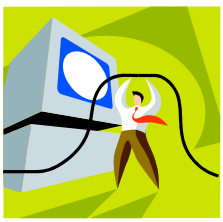


ÖĞRENME FAALİYETİ–1

AMAÇ

Programın temel parçalarından olan veri giriş ve çıkış işlemlerini yapabileceksiniz.

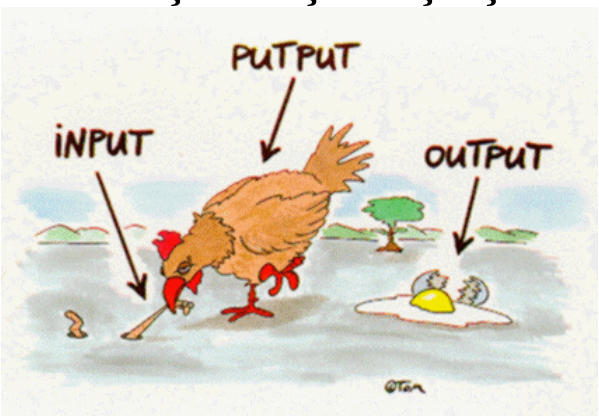
ARAŞTIRMA



Bu faaliyet öncesinde hazırlık amacıyla aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Programlama dillerinin Windows, Linux ve Macintosh üzerinde alternatif veya uyumlu sürümleri olup olmadığını araştırınız. Sonuçları rapor haline getirip arkadaşlarınızla paylaşınız.
- Bilgisayarınızda bilgi girişi ve çıkışı için hangi donanımları kullanıyorsunuz? Mesela, dokunmatik ekran ve ışık kalemini (*stylus*) araştırınız.

1. VERİ GİRİŞ VE ÇIKIŞ İŞLEMLERİ



Hangi dili kullanırsanız kullanın (C, Basic, Perl, Pascal, Java…) tüm dillerde belli konularda aynı prensipler vardır. Bu temel konuları öğrenirseniz, çoğu dillerde rahatlıkla aynı işlemi yapabilirsiniz.

Burada genellikle basit dillerden örnekler verilecektir. Doğrudan Java veya C# gibi dillere geçmek, yüzmeyi öğrenmeden köpek balıkları ile dolu okyanusu aşmaya benzer. Modülde yeri gelince bu dillerden de birkaç örnek verilecektir.

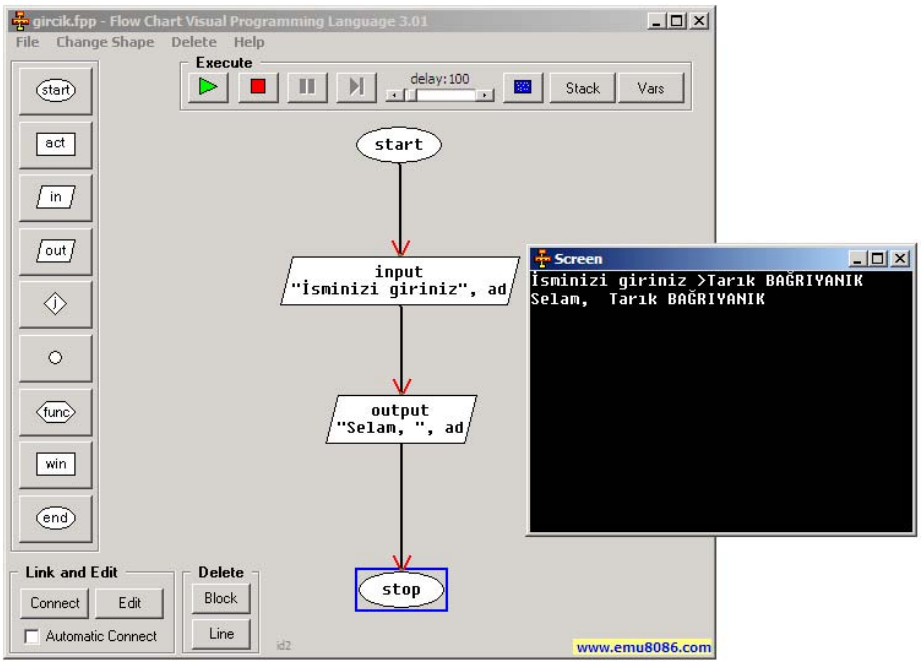
1.1. Giriş ve Çıkış Deyimleri

Bilgisayar; temel olarak verinin girilmesi, işlenmesi ve çıktı olarak kullanıcıya verilmesi işlemlerini yapar. Veriler bilgisayarın anlayacağı şekilde girilirken, bilgisayar da kullanıcının anlayacağı şekilde sonuçların çıkışını verir. Veri doğru girilirse işlemler doğru yapılır.

Veri giriş ve çıkışı konusunda basit bir algoritma ile örnek yapalım. Klavyeden adınızı girip, ekrana "Selam, " yazısı ile adınızı gösterelim:

1. Başla
2. Metin AD
3. Yaz; "İlk Program"
4. Yaz; "İsminizi giriniz"
5. Oku; AD değişkenine* ata
6. Yaz; "Selam, " & AD**
7. Bitir

Sıra No	AD	Çıkış
1	Ali	**Selam, Ali**
2	Ahmet	**Selam, Ahmet**
3	Ayşe	**Selam, Ayşe**



Resim 1.1: Giriş çıkış programımızın akış şeması

Akış şemasını üstteki "**Flow Chart**" isimli program ile yapmak zorunda değilsiniz. Bir kâğıt üzerine düzgünce planınızı çizebilir, çalışmasını takip ederek test edebilir ve emin olduktan sonra programlama diline kodlamasını yapabilirsiniz. Akış diyagramlarını daha iyi kavramak için bol alıştırma yapınız.

Çok karmaşık akış diyagramlarını adım adım, kendi vereceğiniz değerler ile denemeyi unutmayınız. İstenmeyen durumları denetim altına alınız. Değişkenlerin değerlerini takip etmenin zor olabileceği durumlarda değerleri bir tabloda tutabilirsiniz. Test değerlerinden elde edilen değerleri de ayrı bir tabloya aktarmanız faydalı olacaktır.

* Değişkenler sayesinde geçici bilgileri bellekte saklar ve üzerlerinde işlem yapabiliriz.
** Genellikle + veya & simgesi metinleri birleştirmek, yan yana yazmak için kullanılır.

Güzelce tasarlayıp test ettiğiniz akış şeması ve sahte kodlarını, diğer programcıların da rahat anlaması için yardım belgeleri oluşturabilirsiniz. Yani belli kritik yerlere "açıklama satırları" *** ekleyebilirsiniz. Program güncellenirken, siz orada bulunmasanız bile, diğer programcılar rahatlıkla sizin yerinize işleri yürütebilmelidir. Böylece takım arkadaşlarınız arasında da bir bağ kurulmuş olur. Ayrıca benzer programlarda da aynı akış şeması kullanılabilir. Tekrar tekrar benzer şemaları çizmeye gerek kalmaz.

? Aşağıdaki örnek programı; algoritma ve akış şeması ile karşılaştırınız.

Basic dilinde giriş çıkış örneği

```
PRINT "İlk Basic Programımız"
PRINT "İsminizi giriniz "
INPUT AD$
PRINT "Selam, " + AD$
END
```

Not: Türkçe karakterler problem çıkarabilir diye "ç, ğ, ı, ö, ü, ş" harfleri yerine "c, g, i, o, u, s" kullanıyoruz. Aynı problem web sayfalarında da karşımıza çıkabilir. Bu problemin nedenini araştırınız.

Mikrodenetleyicili (PIC, 8051…) sistemlerde ise veri girişi düğme (*button*) ve algılayıcılar (*sensor*) ile yapılırken; çıkış motor, led, gösterge (*display*) gibi nesneler ile elde edilir.

Labcenter ISIS Proteus ve benzeri programlar ile bilgisayarınızdan elektrik, elektronik deneyleri yapabilirsiniz. Kaynak kodunuzu yazıp, aynı ara birimde derleyip, çalışmasını test edebilirsiniz. Bu program sayesinde hem dijital hem de analog devreler tasarlanabilir. Çeşitli ölçü aletleri ile de devrenin çeşitli noktalarındaki değerler ölçülebilir. Devrenin çalıştığından emin olduktan sonra aynı firmanın Ares programı ile baskı devre kısmını düzenleyebilirsiniz.

Programların her zaman klavyeden veri girişi olmasa da, bir şekilde veri alışı, onu işlemesi ve kullanıcının anlayacağı biçimde çıkışı vardır. Eski programlarda veri giriş çıkışı işlemleri, çoğu kez klavye, ekran ve yazıcı olarak yapılabilirdi. Aslında klavye dışında fare ve dosyalar da veri girişi için kullanılırlar. Modem'den gelen bilgi ve tarayıcıdan gelen resim de giriş olarak kullanılabilir. Ses kartından hoparlöre gönderilen ses de çıktı olarak nitelendirilir..

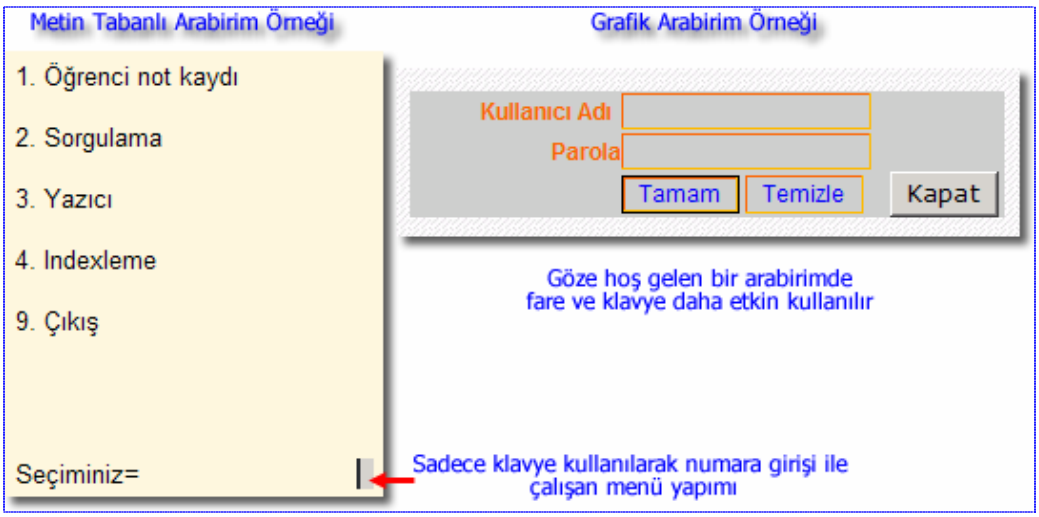
1.2. Ara Birim

"*Character User Interface* (**CUI**)" yani metin tabanlı kullanıcı ara birimi ile çalışmak pek eğlenceli ve kolay gözükmüyor. Bu sebeple programcılar* görsel olarak pencereler, düğmeler, resimler ile programlama yapabilmemiz için "**görsel programlama dilleri**"ni

*** Açıklama satırları; derleyici tarafından derlenmeyen, kodları ve alt programları açıklayıcı metinlerdir. Açıklama satırının önüne ' (tek tırnak), // gibi semboller konulur.

* İlk grafik ara birimi, Xerox şirketi tarafından 1970'li yıllarda yapılmıştır, fakat 1980'lerde Apple kullanana dek grafik ara birim kullanımı yaygınlaşmamıştır.

yazmışlardır. Böylece "*Graphical User Interface* (**GUI**)" denen grafiksel kullanıcı ara birimi deyimi ortaya çıkmıştır. Örneğin; bankaların otomatik para çekme makinelerinde artık sadece rakam girişi değil, ayrıca dokunmatik ekran kullanarak renkli menüler aracılığı ile işlemler yapılabilmektedir.



Resim 1.1: CUI ve GUI karşılaştırması

Metin tabanlı ara birimde genellikle "basit" olarak giriş yapılabilen bir ana menü vardır. İstenen menü elemanın numarası yazılıp "Enter" tuşuna basıldıktan sonra veri girişi, listeleme gibi işlemler yapılır; işlem tamamlanınca da ana menüye dönülür. Grafik ara biriminde de benzer olarak kullanıcı giriş yapana kadar genel bir görsel ara birimde beklenir. Fare ile istenen yer tıklandıktan veya bir klavye kısa yoluna basıldıktan sonra diğer işlemler yapılır. Mesela "Dosya" menüsünden "Aç" komutu verilmesi gibi.

Örnek bir veri giriş çıkış programı:

1. Başla
2. Metin Anne, Baba
3. Yaz; "Anne ve baba adını giriniz"
4. Oku; Anne, Baba
5. Yaz; "Anne adı" & Anne
6. Yaz; "Baba adı" & Baba
7. Bitir

Klavye ve ekranın birleşmesine "**konsol**" denir. Hâlâ modern programlama dilleri "Console Application" proje türünü sunmaya devam ediyorlar. Görsel bir arabirime gerek duyulmuyor ve az yer kaplayan program istenmiş ise, bu tür uygulamalar tercih edilmektedir.

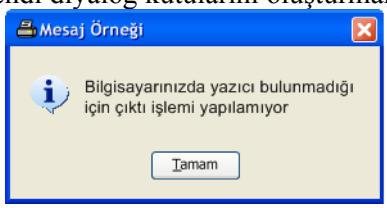
? Araştırınız: Cep telefonlarının gelişimini inceleyiniz. İlk cep telefonları ile son cep telefonlarının ara birim, veri giriş ve çıkış yöntemlerini karşılaştırınız.

Birçok modern işletim sistemi pencereli (**masaüstü** de denir) ara birime sahiptir. Ayrıca programlama dillerinin çoğu görsel sürümlerini piyasaya sürmüşlerdir. C dilini Borland firması CBuilder, Microsoft firması Visual C olarak çıkarmıştır. Pascal dilinin görsel haline Delphi, Basic dilinin görsel haline de Visual Basic diyoruz. Eğer bilgisayarınız çok eski değilse büyük bir ihtimalle görsel programlama dilleri ve web programcılığı ile ilgileniyor olacaksınız.

? Araştırınız: Web sayfalarında kullanıcı ile nasıl iletişim kuruluyor? İçinde arama kutusu bile olmayan bir sitede ne kadar süre kalıyorsunuz? En çok ziyaret ettiğiniz, beğendiğiniz sitelerden derleme yaparak arkadaşlarınız ile paylaşınız.

1.3. Diyalog Kutuları

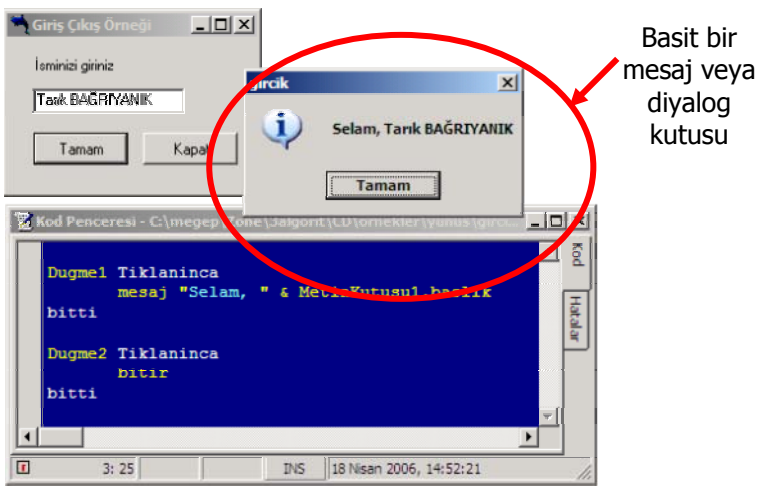
Programcıların basit giriş çıkış işlemlerini halletmeleri için sistemde tanıtılmış hazır pencereler vardır. Bu işlemler **diyalog kutuları** ile yapılabilmektedir. Standart diyalog kutuları her ne kadar hazır form olarak bizim yardımcımız olsa da, programa tam hâkim olmak isteyen programcılar kendi diyalog kutularını oluşturmaktadırlar.



Resim 1.2: Bir mesaj örneği

```
Yunus dilinde giriş çıkış örneği
Dugme1 Tiklaninca
       Mesaj "Selam, " & MetinKutusu1.baslik
Bitti

Dugme2 Tiklaninca
       Bitir
Bitti
```



Resim 1.3: Yunus ile yapılan giriş çıkış programımızın çalışma anı

Sık sorulan bir soru:

Ara Birim nedir?

Cevap:

İki bağımsız sistemin birbirleri ile anlaşmasını sağlayan, hatalı veri girilmemesi için sınırlayıcı ortamdır. Bilgisayar teknolojisinde birkaç çeşit ara birim vardır:

- **Kullanıcı ara birimi:** Kullanıcı ile işletim sistemi arasında, klavye ve fare ile etkileşim sağlar.
- **Yazılım ara birimi:** Uygulamaların kendi aralarında ve donanım ile olan iletişimini sağlayan yazılımlardır.
- **Donanım ara birimi:** Donanım elemanları arasında iletişim sağlayan soket, kablo gibi parçalardır.

1.4. Yazıcı

CUI programlama dilleri ile yazıcı kullanımı, hemen hemen ekrana yazı yazma ile aynıdır. GUI programlama dillerinde özel nesneler sayesinde bu zahmetli iş, çok daha kolay hâle gelmiştir. Crystal Reports nesnesi Visual Basic için, Quick Report ve Rave Report da Delphi ve CBuilder için geliştirilmiştir. Bu nesneler ile profesyonel görünümde rapor hazırlayabilirsiniz.

Karakter tabanlı yazdırma işleminde resim, grafik gibi şekilleri basmak çok uzun program yazımına neden olur. Mümkün değil demesek de, gerçekten zahmetli bir iştir. Bir de yazıcınız nokta vuruşlu ise resimlerin görüntülenmesi zaten iyi olmaz. İstediğiniz yazı tipinde çıktı alamayabilirsiniz. Aşağıdaki Basic örneğindeki gibi yazdırma için yapacağınız şey sadece LPRINT komutunu vermenizdir. L harfi "*line printer* – satır yazıcı" kelimesinden alınmıştır.

Basic dilinde yazdırma örneği

```
LPRINT "İlk yazıcı destekli programımız"
LPRINT
LPRINT "İsmim Tarık BAĞRIYANIK"
END
```

Araştırma 1: LPT kelimesi ve **PRN** kelimelerinin ne anlama geldiklerini araştırınız. Bilgisayarınızdaki Aygıt Yöneticisinden **LPT** ile ilgili bilgi alınız.

Araştırma 2: *Whitespace* (**SP** - boşluk), *linefeed* (**LF** – yeni satır) ve *carriage return* (**CR** – satır başı) karakterleri ne işe yarar? **ASCII** kod tablosunda karakter numaralarını bulunuz.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Veri girme komutu ile klavyeden bilgi okuyunuz.	➢ "Flow Chart" programı ile denemeler yapabilirsiniz. ➢ Klavyeden isim, yaş, doğum yılı gibi bilgiler istenebilir.
2. Veri çıkış komutu ile ekrana yazı yazdırınız.	➢ Hem girilen veri ile ilgili hem de diğer yardımcı bilgileri ekrana yazdırabilirsiniz.
3. Bir dış kaynaktan (internet veya tarayıcı) bilgi alınız.	➢ İnternette bir konuda araştırma yapıp, ilgili dosyaları indiriniz. ➢ Tarayıcıdan bir resmi tarayınız. Bir yazıyı kelime işlemci programına aktarınız. ➢ Dijital fotoğraf makinesi ile resimler çekip, bilgisayarınıza aktarınız, görüntüleyiniz.
4. Yazıcıdan çıktı alınız ve ses kartından müzik çalınız.	➢ Yazıcıdan resim ve yazı çıktıları deneyiniz. ➢ Mikrofon ile kendi sesinizi kaydedebilirsiniz.

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; ilk 9 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. Akış diyagramında "giriş – input" ile değişkene değer ataması yaparız. ( )

2. Ekrana bir değişkenin değerini yazmak için … … … … kullanabiliriz?

3. Program akışında tüm ihtimallerin muhakkak işlemleri yazılmalıdır. ( )

4. Programın doğru çalışıp çalışmadığını rastgele değerler vererek denemeliyiz. ( )

5. Açıklama satırlarını her satır için mecburen yazmamız gereklidir. ( )

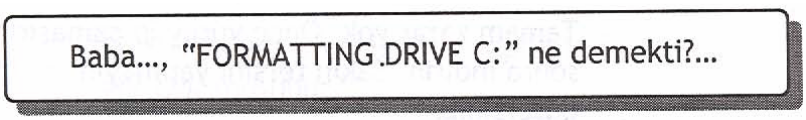
6. Akış şemasını bir kere yaptıktan sonra bir daha hiç kullanmayız. ( )

7. Metin tabanlı kullanıcı ara biriminde fare de kullanılabilir. ( )

8. Metin tabanlı bir sistemde assembly komutları ile grafik ekrana geçilebilir. ( )

9. Her işletim sisteminde kullanıcıya yardımcı basit, kabuk (*shell*) programlama dili vardır. ( )

10. Hangi seçenek hatalıdır?
A) Modem, sadece veri girişi için kullanılır.
B) Eski bilgisayarlarda ses çıkışı, basit bir hoparlör ile elde edilirdi.
C) Tablet bilgisayarlarda, ekrandan veri girişi yapılabilir.
D) Fare, sadece veri girişi için kullanılır.

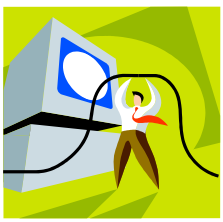


ÖĞRENME FAALİYETİ–2

AMAÇ

Bu öğrenme faaliyetinde, programda temel işlevlerin gerçekleştirilmesi için kullanılan değişken yapısını ve kullanımını öğreneceksiniz.

ARAŞTIRMA



Bu faaliyet öncesinde hazırlık amacıyla aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Bilgisayarda veri girişleri geçici veya kalıcı olarak nerelerde, hangi donanım parçaları içinde nasıl saklanıyor? Mesela oyunlarda kayıtlar nerede tutuluyor veya hesap makinesinde hesaplanan sayılar nerede tutuluyor sorularına cevap arayınız.
- İnsan beyninin depolama kapasitesi ve hızı ne kadardır? Günümüz bilgisayarlarında ortalama olarak ne kadar depolama kapasitesi ve hız değeri vardır? Araştırınız.

2. DEĞİŞKENLER

Temel programlama işlevlerini gerçekleştirebilmek için "**değişkenler**" kullanılmaktadır. Bir değişken küçük bir depo alanıdır. İçinde sayılar, kelimeler, harfler saklanabilir. Değişkenleri defalarca kullanabilsek de, içinde aynı anda sadece **bir** bilgi tutabilmektedirler. Değişkenler geçici olarak kullanıldıkları için, program veya bilgisayar kapanınca silinirler.

Bazı değerler ise her zaman aynı değere sahip olduğu için "**sabit**" adını alırlar. Mesela PI[†] sayısı olan 3.14 değeri gibi. Bu tür sabit olan bazı değerleri programlama dili kendi içinde "hazır" olarak destekleyebilir.

[†] PI'yi 22/7 olarak da alabilirsiniz. PI "π" simgesi olarak günlük hayatta kullanılıyor.

2.1. Değişken ve Sabitler

Bilgisayarın iç kısmında bir donanım olarak bir hafıza olduğunu biliyorsunuz. Bu iç hafıza kullanıcı girişleri ve işlem sonuçları gibi bilgileri saklar. Kullanıcı girişi ve işlem sonuçları her program çalıştırıldığında değişebildiği için bu komutlara "**değişken**" adı verilir. Değişkenin içindeki bilgiye de "**değer**" adı verilir.

Genellikle değişken ve sabitler programın ilk satırlarında tanımlanmaktadır. Programınızda hangi değişkenlerin kullanılacağını programı yazmadan önce planlamanızı tavsiye ediyoruz. Bu aşamaya "**kurulum**" ismi verilebilir.

Değişkenin adı ve ne tür bilgiyi saklayacağı da çok önemli bir konudur. Öğrenmek istediğiniz dilin değişken türlerini iyi öğreniniz. Ayrıca biraz daha ayrıntılı bir konu olan, değişkenlerin hafızadaki kapladıkları alanlar bazı zamanlarda önemli olabilir.

Başka bir önemli nokta, değişken ve sabitlerin adlandırılmasıdır. Bilgisayar sizin değişkenlere ne isim verdiğinizi umursamasa da, hem sizin hem de diğer programcılar için kolay anlaşılacak şekilde isimlendirilme çok önemlidir. İsim ve tür çok önemli olduğuna göre, isim verirken bu ikisini birleştirebiliriz. Bu tür stilde isimlendirmeye "Macar notasyonu[*]" denilmektedir.

Örnek isimlendirmeler:

tYil = **t**amsayı Yıl
mAd = **m**etin Ad
bAskerlik = **b**oolean (mantıksal) Askerlik

Her ne kadar daha sonraki derslerde göreceğimiz "nesne tabanlı programlama" konusunda isimlendirme problemi olmasa da, klasik programcılık ile uğraşan, ya da programlamayı yeni öğrenen birinin, bu notasyonu prensip edinmesinde fayda vardır.

Genel olarak başka **isimlendirme** kuralları vardır:

- İlk harf veya tamamı sayı olamaz. Mesela: "2nciYari" gibi olmaz.
- Değişken ismi içinde "boşluk, TAB, Enter" olamaz. Mesela "Soy Ad" gibi olmaz.
- Büyük küçük harfle yazım[**] fark eder. Mesela: "IlkNot" ile "ILKNOT" farklıdır.
- Dilin anahtar kelimeleri değişken adı olamaz. Mesela: "PRINT" gibi…
- Türkçe ve özel karakterleri kullanmamaya çalışmalıyız. Mesela: "örütBağ" yerine "orutBag" yazılmalı gibi…
- Alt çizgi isimlendirmede kullanılabilir. Mesela: "taban_Ucret" gibi…

[*] "Hungarian Notation" Charles Simonyi isimli bir Macar bilim adamı 1976 yılında geliştirmiştir.
[**] Pascal ve Basic harf büyüklüklerini önemsemez iken, C ve Java gibi dillerde program boyunca aynı şekilde yazılmayı gerektirir.

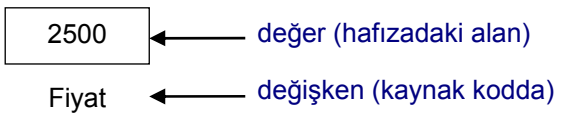
Genel olarak değişken değerleri; **tam sayı**, **ondalıklı sayılar**, **metin**, **karakter** ve **mantıksal** olabilmektedir.

Örneğin; yaşınız **tam sayı** olarak 20, manavdan aldığınız patatesin ağırlığı **ondalıklı sayı** olarak 2.4 kilogram, adınız **metin** olarak "Ayşe", ehliyet türü **karakter** olarak "B" ve son olarak da askerlik durumunuz **mantıksal** olarak "False – Hayır" değer alabilir.

Genel olarak sahte kod yazarken sayılara "**sayısal**", karakterlerden oluşan metinlere "**metin**" denilir.

Değişken isimlendirme ve türünü seçmeden sonra, önemli bir konu da bu değişkenin ilk değerinin ne olacağına karar vermektir. İlk değeri belli olmayan ve program boyunca da değer atanmayan bir değişkenin değeri "**belirsiz**" olarak kabul edilir. Yanlışlıkla da bu değişken, işlemlerde kullanılırsa, programda istenmeyen değerler elde edilebilir. İlk değer atamaları ile böcek oluşmasını en başta engelleyebilirsiniz.

Bir başka sık oluşan böcek de, değişken ismini program ilerledikçe unutarak ya da klavyeden yanlış tuşlara basmaktan dolayı, farklı yazmaktır. Eğer en başta "OkulAdi" diye değişken tanımlamış, daha sonra da "OkulunAdi" diye değişkeni kullanmaya başlamış iseniz; programınızda bir şeyler garip gitmeye başlayabilir. Değişkenin adı ile ilgili global bir ifade vermeniz çok önemlidir.



Bir değişkenin bellekteki ve kaynak koddaki hali

2.2. Atama

Bir değişkene doğrudan veya kullanıcı girişi ile değer atanabilir. Doğrudan değer atama "=" ile yapılır ve yönü sağdan sola (←) doğrudur. Çoğu programlama dilinde atama "=" ile yapılır. Tupol dilinde "<=" ile, Pascal dilinde de ":=" şeklinde yapılmaktadır.

```
a = 5                  //Doğru; a'nın değeri 5 olur
a = a + 5              //Doğru; a'nın eski değerine 5 eklenir
a + 5 = a              //Hatalı!
5 = a                  //Hatalı!
a = a                  //Gereksiz
```

Genellikle `a = a + 1` gibi komutları, bir şeyleri saymak için sayaç yaparken kullanırız. Bu satır her çalıştırıldığında "`a`"nın değeri 1 artırılır.

Daha sonra göreceğimiz döngüler ile de, bir şeylerin toplamını bulmak istediğimizde de `toplam = toplam + sayi` şeklinde ifadeler kullanırız. "`toplam`" değişkenin eski değerine yeni "`sayi`" değeri eklenerek yeni "`toplam`" değişkenine atanır. Böylece `toplam = 1 + 2 + 3 + 4 …` şeklinde yazmamıza gerek kalmaz.

Faktöriyel alırken de + yerine * işlemi kullanılmaktadır. `faktor = 1 * 2 * 3 * 4` … yerine döngü içinde `faktor = faktor * sayi` ifadesi kullanılır.

```
a = 4                    //a'nın değeri 4
a = 5                    //a'nın değeri 5 oldu, eski değeri olan 4 silindi
```

Dillerin bazılarında daha tanımlama sırasında değer ataması yapılmaktadır. Buna "**varsayılan değer atama**" denir:

```
Dim a As Integer = 5          //Basic dilinde
int a = 5;                    //C dilinde
Const a = 5;                  //Pascal dilinde
```

Bir değişkene değer aktarmanın yolları şunlardır:

- Sabit bir değer aktarmak (ilk değer ataması gibi)
- Bir hesaplamanın değerini aktarmak (bildiğimiz 4 işlem ile)
- Kullanıcıdan değer girmesini istemek (önceki derslerden hatırlayınız)

Değişkenlere değer girilmesi ile ilgili basit bir örnek yapacak olursak, bilgilerimizi pekiştirmiş oluruz.

Örnek 1:

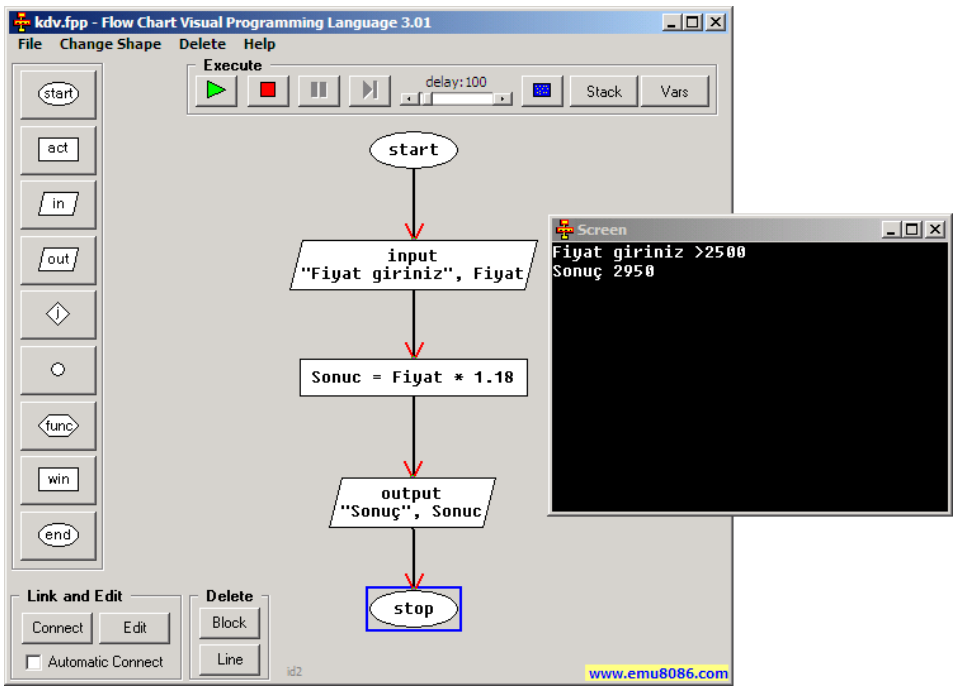
Klavyeden iki sayı girilir, yer değiştirilerek ekrana yazdırılır:

1. Başla
2. Sayısal Sayı1, Sayı2, Değiştirici
3. Oku; "İki sayı giriniz", Sayı1, Sayı2
4. Değiştirici = Sayı1
5. Sayı1 = Sayı2
6. Sayı2 = Değiştirici
7. Yaz; Sayı1, Sayı2
8. Bitir

Örnek 2:

Klavyeden bir fiyat girilir, KDV eklenerek sonuç değeri ekrana yazdırılır:

1. Başla
2. Sayısal Fiyat, Sonuc
3. Yaz; "Fiyat giriniz"
4. Oku; Fiyat
5. Sonuc = Fiyat * 1.18
6. Yaz; Sonuc
7. Bitir

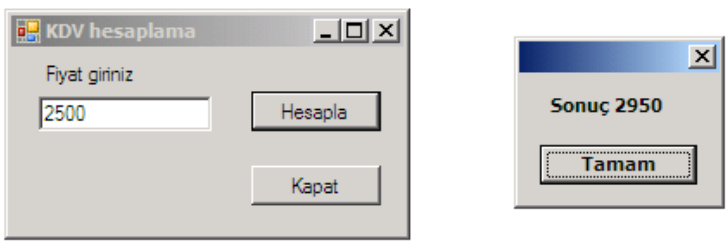


Resim 2.1: Değişkenler ile oynamak

? Aynı programın VB dilinde yazılmış halini inceleyiniz.

```
    Private Sub Button2_Click(ByVal sender As System.Obje
        End 'Programı sonlandırır
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Obje
        fiyat = TextBox1.Text
        sonuc = fiyat * 1.18
        MessageBox.Show("Sonuç " & sonuc)
    End Sub
End Class
```



Resim 2.2: KDV hesaplama programı

Atamalar ile ilgili hem kullanıcı girişi, hem de hesaplama yapılarak değer aktarmayı gerçekleştirdik.

Değişkenlere sabit değer ataması önemli olan ikinci bir konudur. Sabit değer atamasının faydasını şöyle açıklayabiliriz. KDV değerinin 1.18 olduğunu düşünelim, bir gün bir değişiklik olduğunda sabit tanımladığınız rakamı bir kez değiştirirsiniz ve tüm programda bu yeni değer geçerli olur. Bu yöntem tüm programı tarayıp değerleri güncellemekten daha iyidir. Değer güncellemede zaman kaybı olmaması ve hata oluşmaması için gereken yerde sabit kullanabiliriz.

"Sabit atama" yöntemi ile örnek yapalım:

1. Başla
2. Sayısal KDV, Fiyat, Sonuc
3. KDV = 1.18 //Tüm programda hep aynı değer olarak geçerli
4. Yaz; "Fiyat giriniz"
5. Oku; Fiyat
6. Sonuc = Fiyat * KDV
7. Yaz; Sonuc
8. Bitir

Sıra No	Fiyat	KDV	**Sonuc**
1	500	1.18	**590.0**
2	330	1.18	**389.4**
3	300	1.18	**354.0**

Metinler ve sayılar üzerinde birçok işlem yapabiliriz. Metinleri birleştirebilir, arasından bazı kelimeleri alabiliriz, silebiliriz, değiştirebiliriz. Sayıları da çarpabilir, bölebilir, sinüsünü alabilir, logaritmasını alabiliriz. Bütün bu işlemler değişken ataması yapılarak gerçekleştirilir.

Genellikle basit işlemleri "hesap makinesi[‡]" ile yapıyoruz. Biraz uzun, karmaşık formüllerde "hesaplama ve tablolama" programları kullanabiliriz. Matematiksel ve bilimsel işlemleri çözmek için yazılmış birçok program da bulunmaktadır.

2.3. Açıklama Satırları

Modüldeki bazı örneklerin belli yerlerinde kod haricinde, kendi dilimizde cümleler halinde açıklayıcı bilgiler bulunmaktadır.

Küçük programları çoğu programcı açıklama satırları olmadan çözebilir. Ama biraz program büyüdüğünde hangi satırın ne işe yaradığı unutulabilmektedir. Satırları tek tek tarayıp ne işe yaradıklarını çözmek yerine makul açıklama bilgileri ile programcıya yardım etmek gereklidir. Nitekim biz de yaptığımız programı daha sonradan incelediğimizde hatırlamayabiliriz.

[‡] "Başlat*Çalıştır…*calc" ile hesap makinesini açabilirsiniz

Açıklama satırlarının kullanım amaçları:

- Programı yazan hakkında bilgi vermek
- İlk oluşturma tarihi ve güncelleme tarihi bilgisini vermek
- Programın amacının ne olduğu bilgisini vermek
- Problemi nasıl çözdüğü bilgisini vermek
- Programa giren, kaydolan ve çıkan bilgilerini vermek
- Programdaki hatalar hakkında bilgi vermek
- O anda işe yaramayacağı düşünülen satırı iptal etmek
- Gizlenmiş böcekleri bulmak için kodların belli yerlerini iptal etmek

Açıklama satırları çeşitli simgeler ile yapılır. Her programlama dilinde açıklama satırı yapılabilir. Örneğin, C ve Delphi dilinde // iki bölü işareti, Basic dilinde ' tek tırnak kullanılır.

```
Kod1;
//açıklama satırı
Kod2;
```

Genellikle programcı kod satırlarının başına açıklama satırı işareti yazar. Bazı dillerde ise satırın bir kısmını açıklama satırı hâline getirme imkânı da vardır. Pascal dilinden örnek:

```
Kod1; Kod2; (* açıklama satırı *) Kod3;
```

Not: Açıklama satırları derlenen programın boyutlarını artırmaz. Derleyici açıklama satırlarını göz ardı eder. Yine de açıklamalar hikâye şekline dönüşecek kadar uzatılamamalıdır.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Kısa bir program yapınız.	➢ Mesela girilen sayılar üzerinde basit matematiksel işlemler yapan bir program olabilir. ➢ Programınızdaki değişken adını belirlerken, hatırlatıcı şekilde olmasına dikkat ediniz. İsimlendirme işlemini uydurma yöntemi ile değil, ne ile ilgili olduğunu hatırlatma yöntemi ile yapınız.
2. Tanımlama yazımında ilk değeri veriniz.	➢ Değişkenlerin varsayılan bir değeri olmalıdır.
3. Değişken türünü yazınız.	➢ Değişkenin türünü en başta karar veriniz, sonradan değiştirince, istenmeyen sonuçlar oluşabilir.
4. Değişkenler üzerinde matematiksel ve metin işlemleri yapınız.	➢ "Flow Chart" programı ile denemeler yapabilirsiniz.
5. Gerekli yerlere açıklama satırları yazarak, programcıya bilgilendirme yapınız.	➢ Uzun açıklamalardan kaçınınız. Uzun anlatmak yerine hiç açıklama yapmamak daha iyidir.
6. Sonucu, ekranda kullanıcının anlayacağı şekilde görüntüleyiniz.	

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; ilk 10 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. Sabitlere sonradan değer aktarılamaz. ( )

2. Tüm dillerde, değişkenleri programın istediğimiz yerinde tanımlayabiliriz. ( )

3. Değişkenin ismini istediğimiz gibi verebiliriz. ( )

4. Bir değişkenin ilk değerini her zaman boşluk olarak atamalıyız. ( )

5. Atamalar çift tırnak içine yazılır. ( )

6. Kullanıcı girişlerini kontrol etmeden işlemlerde kullanabiliriz. ( )

7. Metin değişkenler üzerinde her türlü matematik işlemi yapabiliriz. ( )

8. Metin değişkenleri birbirinden çıkarabiliriz. ( )

9. Bir programda istediğimiz kadar değişken tanımlayabiliriz. ( )

10. Açıklama satırları programın çalıştırılabilir dosyasının boyutunu artırır. ( )

11. Programdaki hataların kaynağını bulabilmek için açıklama satırları ile o kısmı kapatabiliriz. ( )

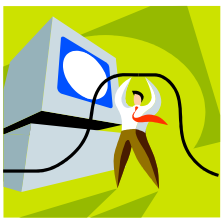
12. Hangi değişken isimlendirmesi <u>hatalıdır</u>?
A) A5
B) _B
C) ölçü birimi
D) AntAlya

ÖĞRENME FAALİYETİ–3

AMAÇ

Programın temel parçalarından olan sayı ve metin işlemlerini yapabileceksiniz.

ARAŞTIRMA



Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Metin ve sayı değişken olarak tanımlayabileceğiniz neler vardır? Mesela televizyondaki bir kanalın adı ne tür, kanalın numarası ne tür olabilir? Diğer örnek soru; adresinizdeki hangi bilgiler yazı türünde, hangi bilgiler sayı türündedir?
- Mesajlaşma programlarında kelimelerin kısaltmalarını nasıl kullanıyorsunuz? Anlamlı olarak kelimeleri en kısa nasıl yazabilirsiniz?

3. SAYI VE METİN TÜRLERİ

Sayılar[§] ve metinler en temel değişken türlerimizdir. Sayılar ile dört işlem yapabilir, metinler ile de karakterlerden meydana gelen değerleri işleyebiliriz. Metinler genellikle birden fazla karakterden oluştuğu için çift tırnak " " içine almamız gerekir ve içine istediğinizi yazabilirsiniz: sayı, ondalıklı sayı, Türkçe harfler (ç ş ğ ü ö ı) içeren yazı, özel karakterler (! ^ % & / * ~ @ gibi), boşluk…

Matematik işlemler ile ilgili örnekler:

Matematik İşlem	İşleç	Örnek	Sonuç
Toplama	+	2 + 4	6
Çıkarma	–	3 – 4	–1
Bölme	/	4 / 2	2
Üs alma	^	2 ^ 3	8
Mod (kalan) alma	%	5 % 2	1

3.1. Temel İşleçler

İşleç ya da diğer ifade ile operatörler bizim hesaplama işlemlerini yapmamızı sağlayan simgelerdir. Genel olarak şu kategorilere ayrılırlar:

- **Aritmetik işleçler:** `+ - * / ^ %`
- **İlişkisel işleçler:** `= > < <> >= <=`

[§] Programlarda genellikle onluk (*decimal*), onaltılık (*hexadecimal*) ve üstel (*exponential*) sayılar kullanılır. Bilgisayarın iç kısmı ise ikilik (*binary*) sayı sistemi ile çalışır.

- **Mantıksal işleçler:** `Ve(And) Veya(Or) Değil(Not)`

Aritmetik işlemler toplama +, çıkarma –, çarpma *, bölme /, üs alma ^ ve mod[**] alma % olarak genellenebilir. **İlişkisel** işleçler ise değer karşılaştırmalarında kullanılır. **Mantıksal** işleçler ise temel olarak "True – Evet" ve "False – Hayır" mantıksal değerlerini alabilen değerler üzerinde işlem yapmamızı sağlar.

Örnek:

Girilen sayıların modunu alan programı yazalım, deneme değerlerini bir tabloya aktaralım:

1. Başla
2. Sayısal sayı1, sayı2, kalan
3. Oku; sayı1, sayı2
4. kalan = sayı1 % sayı2
5. Yaz; kalan
6. Bitir

Sıra No	sayı1	sayı2	kalan
1	5	3	**2**
2	33	22	**11**
3	3	5	**3**

Günlük hayatta çarpma ve bölmeyi başka simgeler ile gösteririz. Mesela çarpma "." veya "hiçbir şey", bölme de "÷" kullanılır. Bu simgeler ile istenilen sonucu bilgisayarda elde edemezsiniz:

sonuc = ab *a ile b çarpılır, sonuc değişkenine aktarılır*
sonuc = a . b *a ile b çarpılır, sonuc değişkenine aktarılır*
sonuc = a ÷ b *a b'ye bölünür, sonuc değişkenine aktarılır*

Örnek 1:

sonuc = a + b – c + 2abc – 7 ifadesinin bilgisayar dilindeki karşılığı:

```
sonuc = a + b - c + 2 * a * b * c - 7
```

Örnek 2:

$sonuc = \frac{a^2 + b^2}{2ab}$ ifadesinin bilgisayar dilindeki karşılığı:

```
sonuc = (a ^ 2 + b ^ 2) / (2 * a * b)
```

[**] Mod, bölme işleminde tam sayı olarak **kalanı** veren işlem türüdür.

Ayrıca karşılaştırma işleçlerinde "<=" ve ">=" yerine "=<" ve "=>" kullanamazsınız. Üs için de a^b şeklinde değil, a^b kullanınız. Zaten programlama dillerinde üst simge yazılamaz.

İşleç	Tanım	Örnek
Toplama (+)	İşlenenleri toplar.	`x = y + z`
Mod alma (%)	Tam sayı olarak kalanı verir.	`x = y % z`
Eşitlik (=)	İşlenenlerin eşitliğini test eder, eşitlerse True – Evet döner.	`x = y`
Eşit Değil (<>)	Değerler eşit değilse True – Evet döner.	`x <> y`
Ve (And)	İki şartın mantıksal "Ve" işlemine tutulması.	`(x = 3) Ve (y <> 4)`
Değil (Not)	Şartın mantıksal "Değil"ini alma.	`Değil (x = 3)`

Örnek:

Eğer x'in değerini 2, y'nin değerini 10 kabul edersek:

- `(x < 5) Ve (y > 10)`
- `(2 < 5) Ve (10 > 10)`
- `Evet Ve Hayır`
- **`Hayır`**

- `(x < 5) Veya (y > 10)`
- `(2 < 5) Veya (10 > 10)`
- `Evet Veya Hayır`
- **`Evet`**

- `Değil (x < 5)`
- `Değil (2 < 5)`
- `Değil Evet`
- **`Hayır`**

? Araştırınız: And (Ve), Or (Veya), Not (Değil) ve Xor (Özel Veya) işleçlerinin doğruluk tablolarını bulunuz.

3.2. Değişkenleri Kullanmak

Herhangi bir toplama, çıkarma işleminin sonucu, aşağıdaki örnekteki gibi, değişkenlerde saklanır:

```
KDV_Degeri = 2500 * 1.18
```

Sonuç; işlemci tarafından hesaplanarak "KDV_Degeri" değişkenine depolanır. Tabii burada her program çalıştırıldığında hep aynı değer olan 2950 rakamı hesaplanır. 1.18 rakamının genel olduğunu biliyoruz, öyleyse fiyat yerine ve KDV için de bir değişken adı vererek, her defasında klavyeden yeni değer girerek başka sonuç elde edebiliriz:

```
KDV_Degeri = Girilen_Fiyat_ Degeri * KDV_Sabiti
```

Buradaki formülümüz çok basittir. Zamanla birden fazla işlece sahip matematik işlemlerimiz olacaktır. İşleçlerin sayısı arttıkça bilgisayarın doğru işlemi yapması ve bizim rahat anlamamız için **( )** parantezleri kullanmamız gerekebilir.

```
VergiMiktari = Eski_Vergiler + (Gelir * Vergi_Sabiti)
```

Örnek††:

```
Başla
     Sayısal satışFiyatı, indirimliFiyat
     Oku; "Satış fiyatını giriniz", satışFiyatı

     indirimliFiyat = satışFiyatı * 0.95
     Yaz; "%5 indirimli fiyat ", indirimliFiyat
Bitir
```

3.3. İşlem Öncelikleri

Aritmetik işleçlerin önceliğinde genel olarak şunlara dikkat ediniz:

- Bir formülde eğer varsa, **üs** işlemi ilk önce yapılır.
- Sonra varsa, **çarpma** ve **bölme** yapılır.
- Sonra varsa, **toplama** ve **çıkarma** yapılır.
- Eğer aynı tür işleçler var ise, **sol** taraftaki önce hesaplanır.

Örnek:

1. Adım: Önce üs işlemi yapılır
   ```
   islem = 3 + 4 ^ 5 - 8 / 5 * 7
   ```
2. Adım: Soldaki bölme çarpmadan önceliklidir.
   ```
   islem = 3 + 1024 - 8 / 5 * 7
   ```
3. Adım: Çarpma ilk önceliğe sahip oldu.
   ```
   islem = 3 + 1024 - 1.6 * 7
   ```
4. Adım: Sol taraftaki toplama çıkarma işlemine göre öncelik sahibi
   ```
   islem = 3 + 1024 - 11.2
   ```
5. Adım: son işlem adımı
   ```
   islem = 1027 - 11.2
   ```
6. Adım:
   ```
   islem = 1015.8
   ```

Not: Parantez kullanarak karmaşık görünen bir formülü basit hale getirebilirsiniz. Ondalıklı sayılarda "." nokta kullanmaya dikkat ediniz.

†† Başka bir yüzde bulma yöntemi:

indirimliFiyat = satışFiyatı – (satışFiyatı * 0.05)

3.4. Parantez Kullanmak

İşleçlerin önceliğini değiştirmek için parantez kullanabiliriz.

```
deger = 3 + 4 ^ 5
```

Önce 4 üzeri 5 işlemi yapılır, ancak ya 3 ile 4'ü önce toplamak isteseydik ne yapmamız gerekirdi?

```
deger = (3 + 4) ^ 5
```

Birden fazla parantez kullanabiliriz. Bu durumda da önce içteki parantez yapılır. Parantez kalmayana dek tüm parantezler hesaplanır. Daha sonra da normal öncelikli işlemler yapılır.

Örnek:

1. Adım: Önce en içteki parantez içi yapılır.
   ```
   islem = ((3 + 4) ^ 5 / 3 - 8) / 5 * -7
   ```
2. Adım: Üs ilk önceliğe sahiptir.
   ```
   islem = (7 ^ 5 / 3 - 8) / 5 * -7
   ```
3. Adım: Bölme çıkarmadan daha önceliklidir.
   ```
   islem = (16807 / 3 - 8) / 5 * -7
   ```
4. Adım: Parantez içi yapılır
   ```
   islem = (5602.33333 - 8) / 5 * -7
   ```
5. Adım: Soldaki bölme daha önceliklidir.
   ```
   islem = 5594.33333 / 5 * -7
   ```
6. Adım: Son adım olarak çarpma yapılır.
   ```
   islem = 1118.86667 * -7
   ```
7. Adım:
   ```
   islem = -7832.06667
   ```

❓ Sonuç `-7832.06667` çıktı. Eğer parantezler olmasaydı sonuç ne olacaktı?‡‡

3.5. Hazır Matematik Komutları

4 işlem her matematik problemin çözümünde kullanılabilse de; sinüs, kosinüs, logaritma gibi işlemlerde hazır komut kullanmak daha iyidir. Bu komutlar hemen hemen her dilde aynıdır. Her sayı bazı fonksiyonlar için uygun olmayabilir. Mesela negatif sayıların karekökü alınamaz.

Trigonometrik fonksiyonlarda radyan cinsinden değer girilir, bu sebeple önce değeri dereceden radyana dönüştürmek gereklidir. 1 PI "π"; 180 derece anlamına gelir. **PI** değeri

‡‡ Cevap `355.53333`

genellikle dilin içinde tanımlı bir sabittir. Doğru orantı ile istenilen açının radyan değeri şöyle hesaplanabilir:

```
radyanDegeri = girilenDeger * PI / 180
```

*Not: Sıfıra bölünme hatası yani "**Division by zero**" ile karşılaşmamak için, bölme işleminin olduğu her yerde, önce bölenin sıfır olup olmadığını kontrol ederek, önlem almayı unutmayınız.*

? Sizce hangisi daha güvenilir olarak kullanılıyor: analog mu, dijital mi? Mesela analog tartı aleti mi, dijital tartı aleti mi daha doğru ölçüm yapar?

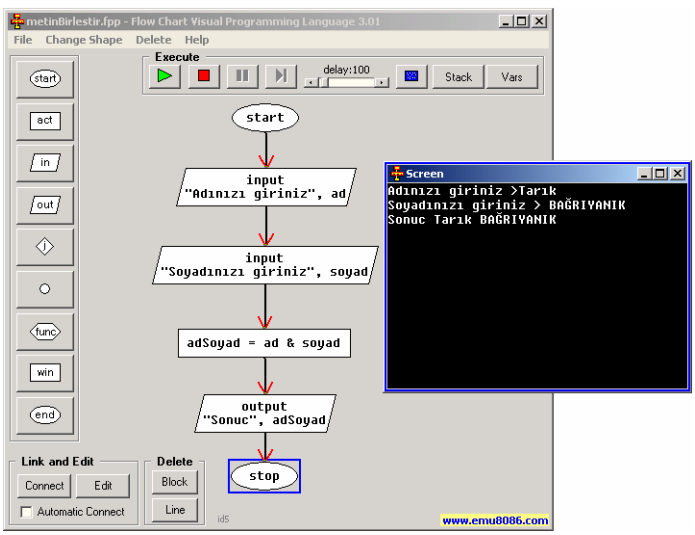
3.6. Metin İşlemleri

Çift veya tek tırnak içine alınan metinlerin içine istediğimizi yazabiliriz. Fakat "3*4" gibi matematik işlemi yazsak, hesaplama işlemi yapılmadığı için, bize sonucunu vermez. Çünkü programlama dili çift tırnak içindeki değeri sayı olarak değil, **metin** olarak algılar.

Metinler + veya & işleci ile birleştirilerek tek bir metne dönüşebilirler. Birleşik olarak yan yana görüntülenmemeleri için aralarına bir boşluk eklemeyi unutmayınız:

```
soyadi = "Bağrıyanık"
adSoyad = "Tarık " & soyadi     //" " boşluk anlamındadır
```

1. Başla
2. Metin ad, soyad, adSoyad
3. Yaz; "Adınızı giriniz"
4. Oku; ad
5. Yaz; "Soyadınızı giriniz"
6. Oku; soyad
7. adSoyad = ad & soyad
8. Yaz; adSoyad
9. Bitir



Resim 3.1: Metinleri birleştiren akış şeması örneği

3.7. Hazır Metin Komutları

Metinler için sayılardaki gibi bazı çok gerekli fonksiyonlar vardır. Mesela; metni büyük harfe, küçük harfe dönüştürmek, boşlukları silmek, uzunluğunu bulmak ve sayısal bir değere dönüştürmek gibi.

? Metin içinde kelime veya harf arama ile ilgili komutları araştırınız.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Tanımlanan değişkenlere sayısal hesaplamalar yaparak değer aktarınız.	➢ Önce basit matematiksel işlemleri deneyebilirsiniz, yaşınızı hesaplamak gibi.
2. Elde edilen değeri ekranda gösteriniz.	➢ Sadece sonucu göstermek anlamlı olmaz. ➢ Mesela: "Yaşım: 16" gibi yazdırınız.
3. Karmaşık matematiksel formüllerde gerekli yerlere parantez ekleyerek, işlem önceliklerini belirleyiniz.	➢ Deneyeceğiniz formülleri hesap makinesi ve hesap tablosu programları ile deneyip, sonucu karşılaştırınız. Güvenilir sonuçlar elde etmeye çalışınız.
4. Metin birleştirme operatörü ile birden fazla metin değeri birleştiriniz.	

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Şıklı sorularda uygun şıkkı işaretleyiniz.

1. İşlemlerde ( ) kullanarak belli işlemlere öncelik sağlarız. ( )

2. Metin türünde olan değişkenlerin içinde Türkçe harfler geçmemelidir. ( )

3. "#4+ $,` *?" şeklinde bir metin değişken içeriği olabilir. ( )

4. Bir sayının karesini almak için aşağıdaki gibi yapabiliriz. ( )

```
sonuc = sayi * sayi
```

5. `a = Değil(4 <> 5)` işleminde a'nın değeri "Evet" olur. ( )

6. `b = 1 Özel Veya 0` işleminde b'nin değeri "Hayır" olur. ( )

7. Hangi işlemin sonucu yanlıştır?

İşlem	Sonuç	
A) 6 – 8	-2	…
B) 3 ^ 2	6	…
C) 7 % 4	3	…
D) 3 / 2	1.5	…

8. Hangi işlemin sonucu doğrudur?

İşlem	Sonuç	
A) (2 ^ 4) – 8	0	…
B) 3 * -2	-6	…
C) ((4 / 2)/ 2)/ 2	2	…
D) 4 ^ (1 / 2)	4	…

9. 30 derece (…) radyandır.

10. Türkiye'nin bölgesel ayarlarında "," virgül ondalık simgesi "." nokta basamak gruplandırma simgesidir. ( )

"İleride bilgisayarlar 1.5 tondan daha hafif olacaklar..."

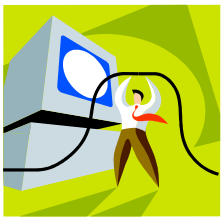
- Popular Mechanics dergisinin 1949 yılına ait bir sayısındaki "Teknolojinin hızlı ilerlemesi"ni konu eden bir yazıdan...

ÖĞRENME FAALİYETİ–4

AMAÇ

Programın temel parçalarından olan kontrol deyimleri ile işlemler yapabileceksiniz.

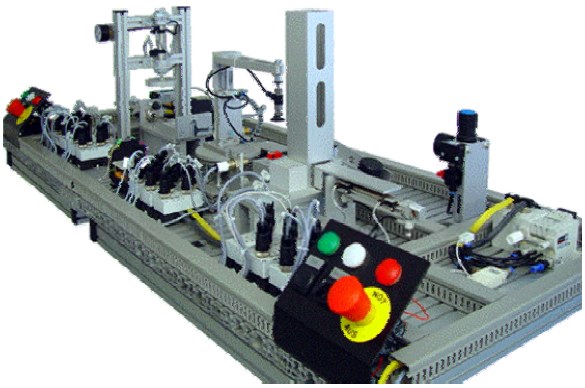
ARAŞTIRMA



Bu faaliyet öncesinde hazırlık amacıyla aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Satranç oyununda, bir taş kaç çeşit hamle yapabilir? Satranç oynamak için geliştirilen programları inceleyiniz.
- Telefon bankacılığında veya internet üzerinden uçak bileti sorgulamada hangi seçenekler kullanıcıya sunuluyor? Arkadaşlarınıza genel olarak işlem adımlarını anlatınız.

4. KONTROL DEYİMLERİ



Şimdiye kadar yapılan örnekler sadece bir kere çalıştırılıyordu, tek seçenekleri vardı. Bizim girdiğimiz veriye göre akışı değişen bir yapısı yoktu. Bu tür programlar genelde pek bir işe yaramaz. Bilgisayara davranış kazandırmanın yolu "**kontrol deyimleri**"ni kullanmaktır. Veri girildikçe çoğu program davranışını değiştirir.

"**If** – Eğer ve **Case** – Durum" gibi komutlar ile programın akışını düzenleriz.

4.1. "Eğer – Değilse" Komutu

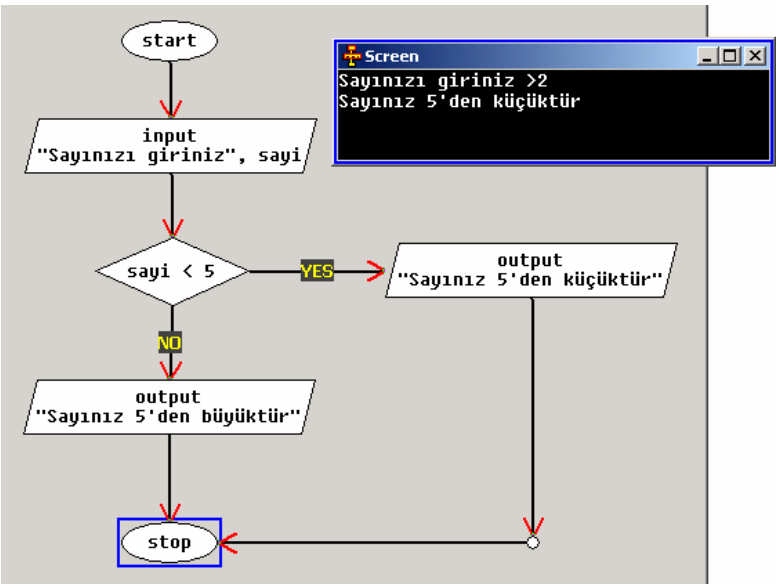
Bir karar vermemiz gerektiğinde soruyu sorar, cevabına göre işlemler yaparız. Bilgisayar da benzer şekilde çalışır. Cevap **True**/**False**; Var/Yok; Evet/Hayır; 0/1 olabilir. Bu değerleri kullanarak problem çözümlenmesine "**Boole**[§§] **Cebiri**" denir.

[§§] George Boole, 19.yüzyılda yaşamış bir İngiliz matematikçidir.

En çok kullanılan komutumuz "**Eğer İse (Değilse)** ya da IF THEN (ELSE)"dir. "Değilse" kısmını kullanmak zorunda değiliz. Ama saçma veya imkânsız da olsa diğer ihtimalde, ne yapılacağını bilgisayara öğretmemiz iyi olur. Daha sonra oluşabilecek hataları baştan engellemiş oluruz.

1. Başla
2. Sayısal sayi
3. Yaz "Sayınızı giriniz"
4. Oku; sayi
5. Eğer sayi < 5 İse
6. Yaz; "Sayınız 5'den küçüktür"
7. Değilse
8. Yaz; "Sayınız 5'den büyüktür"
9. Eğer Bitti
10. Bitir

? Aşağıdaki akış diyagramı programında eğer giriş **5** olursa sonuç ne olur, deneyiniz.



Resim 4.1: Karar deyimi

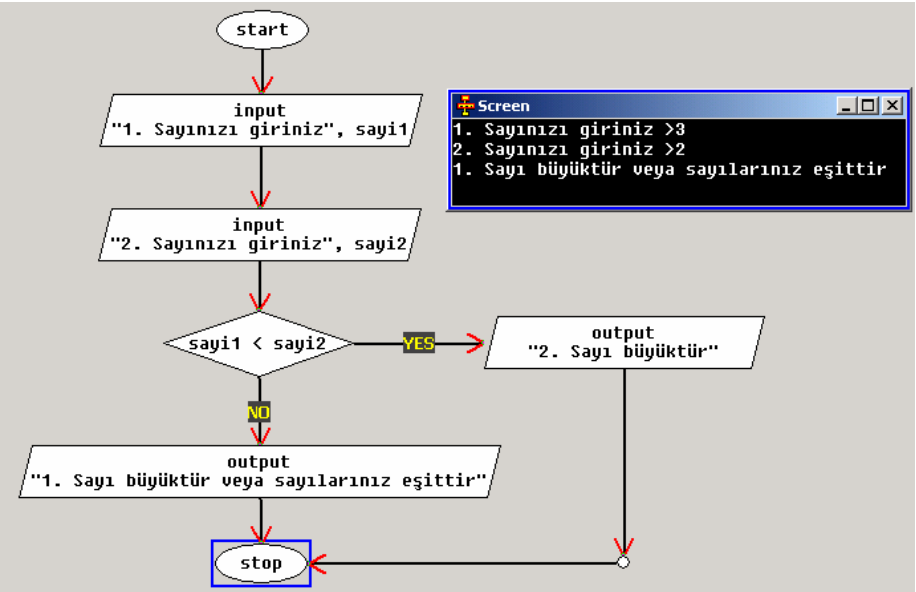
Deyim	Tanım	Değeri
`4 < 6`	4, 6'dan küçük mü?	Evet
`4 > 6`	4, 6'dan büyük mü?	Hayır
`4 = 6`	4, 6'ya eşit mi?	Hayır
`4 <= 6`	4, 6'dan küçük veya eşit mi?	Evet
`4 >= 6`	4, 6'dan büyük veya eşit mi?	Hayır
`4 <> 6`	4, 6'ya eşit değil mi?	Evet

Şart kısmının sonucu sadece Evet – **True** veya Hayır – **False** olur. Bu değeri **boolean - mantıksal** bir değişkene atayabiliriz:

```
sonuc = (4 < 6)
```

Bu işlem sonunda "sonuc" değişkeninde Hayır **- False** değeri atanır. Şart kısmında değişken kullanmak programınıza esneklik sağlar:

```
sonuc = (sayi1 < sayi2)
```



Resim 4.2: İki sayının karşılaştırması

Resim 4.2'deki iki sayının karşılaştırılması örneğinin bloklu sahte kod ile yazımı aşağıdaki gibi yapılabilir.

Örnek 1:

```
Başla
     Sayısal Sayı1, Sayı2
     Oku; "Birinci sayıyı giriniz ", Sayı1
     Oku; "İkinci sayıyı giriniz ", Sayı2

     Eğer Sayı1 < Sayı2 ise
          Yaz; "Sayı1 Sayı2'den küçüktür"
     Değilse
          Yaz; "Sayı1 büyüktür veya sayılarınız eşittir"
     Eğer Bitti
Bitir
```

Örnek 2:

```
Başla
     Sayısal Yaş
     Oku; "Yaş bilgisini giriniz ", Yaş

     Eğer Yaş = 0 ise
          Yaz; "Yaş 0 olamaz"
     Eğer Bitti
     Eğer Yaş < 0 ise
          Yaz; "Yaş negatif olamaz"
     Eğer Bitti
Bitir
```

Örnek 3:

```
Başla
     Sayısal numara, ara
     Oku; "Bir tamsayı giriniz ", numara
     ara = numara % 5

     Eğer ara = 0 ise
          Yaz; "Sayınız 5'e bölünebilir"
     Eğer Bitti
     Eğer ara <> 0 ise
          Yaz; "Sayınız 5'e bölünemez"
     Eğer Bitti
Bitir
```

Örnek 4:

```
Başla
     Sayısal numara, ara
     Oku; "Bir tamsayı giriniz ", numara
     ara = numara % 2

     Eğer ara = 0 ise
          Yaz; "Çift sayı girdiniz"
     Eğer Bitti
     Eğer ara = 1 ise
          Yaz; "Tek sayı girdiniz"
     Eğer Bitti
Bitir
```

Örnek 5:

```
Başla
     Sayısal sayı1, sayı2, ara
     Oku; "İki adet tamsayı giriniz ", sayı1, sayı2
     ara = sayı1 % sayı2

     Eğer ara = 0 ise
          Yaz; "İki sayı tam bölünebilir"
     Değilse
          Yaz; "İki sayı tam bölünemez"
     Eğer Bitti
Bitir
```

İç içe birden fazla eğer şartı da kullanılabilir, örneğin:

```
Başla
     Sayısal ucret, fiyat
     Oku; "Ücret miktarını giriniz ", ucret
     Oku; "Elbise fiyatını giriniz ", fiyat

     Eğer ucret > 1000 ise
        Eğer fiyat < ucret ise
          Yaz; "Elbiseyi almak için yeterli ücretiniz var"
        Eğer Bitti
     Eğer Bitti
Bitir
```

Birden fazla şart var ise "**mantıksal işleçler**" ile tek satırda şartları yazabilirsiniz:

Örnek 1:

```
Başla
     Sayısal ucret, fiyat
     Oku; "Ücret miktarını giriniz ", ucret
     Oku; "Elbise fiyatını giriniz ", fiyat

     Eğer (ucret > 1000) Ve (fiyat < ucret) İse
          Yaz; "Elbiseyi almak için yeterli ücretiniz var"
     Eğer Bitti
Bitir
```

Örnek 2:

```
Başla
     Sayısal not1, not2, not3
     Oku; "Birinci öğrencinin notunu giriniz ", not1
     Oku; "İkinci öğrencinin notunu giriniz ", not2
     Oku; "Üçüncü öğrencinin notunu giriniz ", not3

     Eğer (not1 > not2) Ve (not1 > not3) İse
          Yaz; "Birinci öğrenci en yüksek nota sahiptir."
     Eğer Bitti
     Eğer (not2 > not1) Ve (not2 > not3) İse
          Yaz; "İkinci öğrenci en yüksek nota sahiptir."
     Eğer Bitti
     Eğer (not3 > not1) Ve (not3 > not2) İse
          Yaz; "Üçüncü öğrenci en yüksek nota sahiptir."
     Eğer Bitti
Bitir
```

Örnek 3:

```
Başla
     Sayısal paketFiyatı, paketAğırlığı, mesafe
     Oku; "Paketin ağırlığını giriniz ", paketAğırlığı
     Oku; "Mesafeyi giriniz ", mesafe

     Eğer (mesafe >= 0) Ve (mesafe <= 500) İse
```

```
            paketFiyatı =  paketAğırlığı * 50
        Değilse Eğer (mesafe > 500) Ve (mesafe <= 1000) İse
            paketFiyatı =  paketAğırlığı * 100
        Değilse Eğer (mesafe > 1000) İse
            paketFiyatı =  paketAğırlığı * 500
        Eğer Bitti
        Yaz; "Paket fiyatı " & paketFiyatı
Bitir
```

Mesafe (kilometre)	Kilograma göre fiyat çarpanı
0 ile 500	50
501 ile 1000	100
1001 ve üstü	500

Şart kısmında sadece tam sayı değişken olmak zorunda değildir. Yazı, karakter ve ondalıklı sayılar da kullanılabilir. "Eğer" komutunun olumlu kısmı ve olumsuz kısmı çok satırlı olabilir. Öğreneceğiniz dilin komut bloklarının nasıl yapıldığını iyi anlamanız gereklidir.

Aşağıdaki soruların akış şemalarını çiziniz ve test değerleri ile deneyiniz:

1. Klavyeden girilen not bilgisinin 0 ile 100 arası olup olmadığını kontrol eden programı yapınız.

2. Klavyeden girilen 3 tam sayı değerin en büyük ve en küçük olanını gösteren programı yapınız.

3. Klavyeden girilen bir tam sayı eğer 1 ile 5 veya 7 ile 10 arasında ise, ekrana "Sayınız doğru yerdedir.", değilse "Sayınız yanlış yerdedir." yazınız. Aralık belirten sayılar da aramaya dâhildir.

4. Klavyeden girilen bir sayının negatif, 0 veya pozitif olup olmadığını bulan program yapınız.

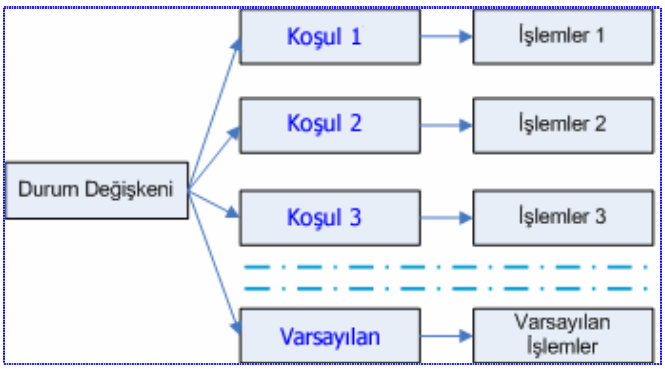
4.2. "Durum" Komutu

Çok sayıda ihtimal varsa ve hepsini "Eğer" komutu ile yaparsanız, program biraz karmaşık hale gelip, takip edilmesi zor olur. Daha derli toplu bir komutumuz var: "**Durum** – Case".

Neredeyse tüm durumları hem liste olarak görürüz, hem de işlemlerini yazarız. Eğer hiçbir ihtimal uymuyor ise, aynen "Eğer" komutundaki "Değilse" gibi "**Varsayılan** – Default" kısmına uygun kodları yazarız.

Tabii her durum tek satırdan oluşmak zorunda değildir, birden fazla komut yazılabilir. Okunaklılığı artırmak için her yeni kod bloğunun biraz daha **içeriden** yazılması iyi olur.

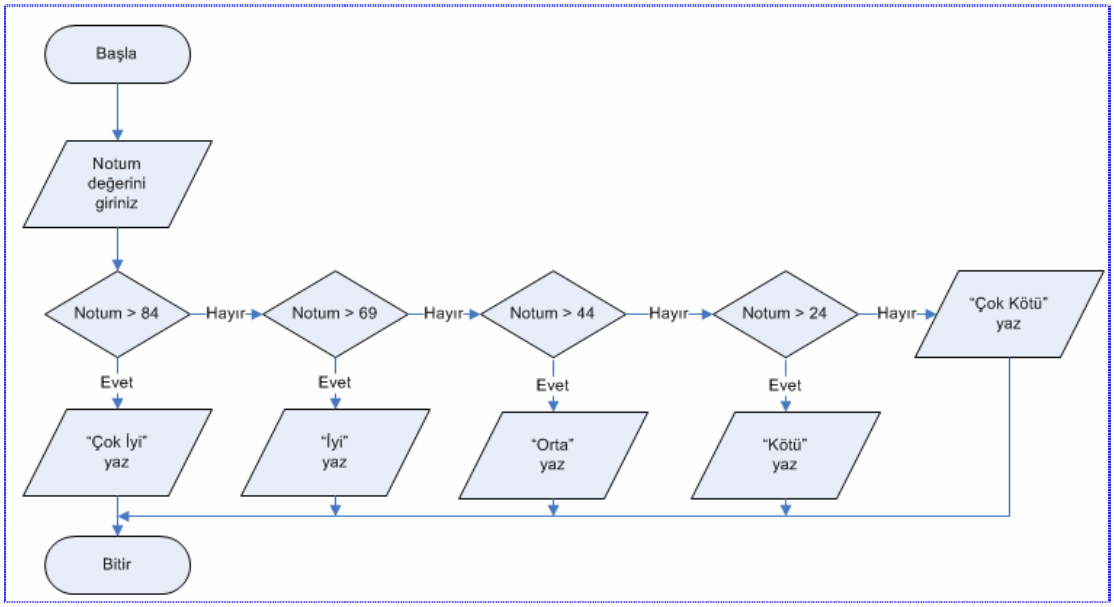
```
...
Durum (değişken)
      Koşul 1:
                  Komutlar
                  Durumdan Çık
      Koşul 2:
                  Komutlar
                  Durumdan Çık
      Koşul 3:
                  Komutlar
                  Durumdan Çık
      Varsayılan:
                  Komutlar
Durum Bitti
…
…
```



Resim 4.3: Durum komutunun görünümü

1. Başla
2. Sayısal Notum
3. Oku; Notum
4. Durum Değişkeni Notum
5. Koşul (Notum > 84)
6. Yaz; "Çok iyi!"
7. Durumdan Çık
8. Koşul (Notum > 69)
9. Yaz; "İyi"
10. Durumdan Çık
11. Koşul (Notum > 44)
12. Yaz; "Orta"
13. Durumdan Çık
14. Koşul (Notum > 24)
15. Yaz; "Zayıf"
16. Durumdan Çık
17. Varsayılan
18. Yaz; "Çok kötü!"
19. Durum Bitti
20. Bitir

"Durum" komutundaki ihtimalleri yazarken birbirini kapsayan ihtimallere dikkat ediniz. Yoksa istenmeyen şekilde programınız çalışabilir.



Resim 4.4: Durum komutunun akış şeması

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Kullanıcıya bir soru sorarak, cevabına göre doğru işlemlere yönlendiriniz, programınızın akış şemasını çiziniz.	➢ Kullanıcıya üst üste soru sorarak çalışan, genellikle "metin tabanlı oyun" adı verilen bir uygulama yapabilirsiniz. ➢ Labirent gibi bir giriş ve çıkışı olan bir ortam düşünebilirsiniz. ➢ Örneğiniz ilerlemek ve çıkışı bulmak için kullanıcıya uygun sorular sorarak çalışacaktır.
2. Girilen veriyi sabit veya hesaplanan değişkenler ile karşılaştırınız.	➢ Kullanıcıya mantıklı sorular sorarak, cevaplarını belli değerler ile karşılaştırınız.
3. Çok değerler veya değer aralıklarında "durum" komutunu kullanınız.	➢ Çok fazla ihtimal var ise "Eğer" yerine "Durum" komutunu kullanınız.

"İnsanların evlerinde bilgisayar bulundurmaları için herhangi bir neden göremiyorum!"

- Ken Olson, Digital Equipment Corp. firmasının kurucusu ve genel müdürü, 1977.

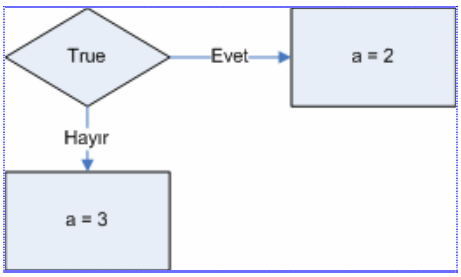
ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

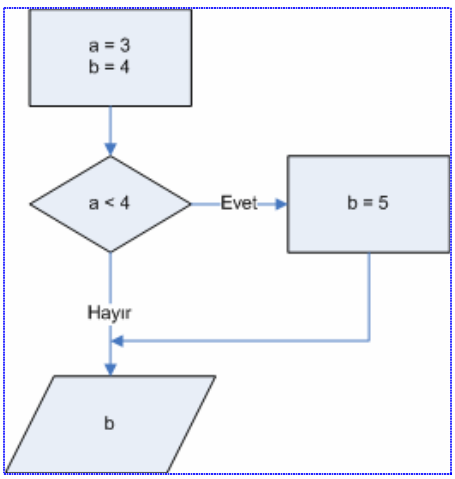
Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Diğer sorularda ise parantez içine uygun cevabı yazınız.

1. "Eğer" komutunda şartın sonucu ya True ya da False olur. ( )

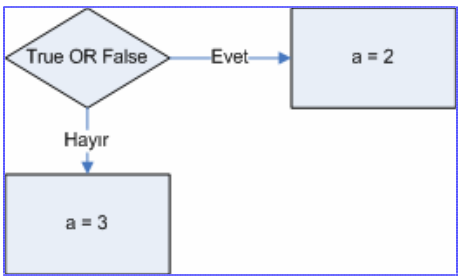
2. Aşağıdaki işleminin sonucunda "a"nın değeri ne olur? (…)



3. Aşağıdaki programda "b"nin değeri ne olur? (…)



4. Aşağıdaki işlemin sonucunda "a"nın değeri ne olur? (…)

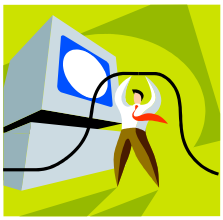


ÖĞRENME FAALİYETİ–5

AMAÇ

Programın temel parçalarından olan döngüler ile işlemler yapabileceksiniz.

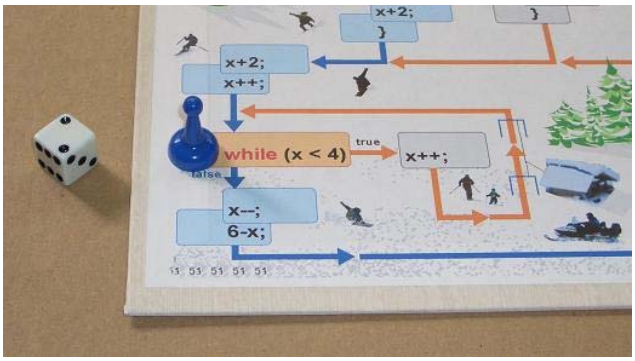
ARAŞTIRMA



Bu faaliyet öncesinde hazırlık amacıyla aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Bilgisayarınızı açtığınızda sık olarak yaptığınız işlemleri sizin için otomatik yapan programlar var mı? Mesela, Opera İnternet Tarayıcısında sık ziyaret ettiğiniz siteleri toplu hâlde açan komut vardır.
- Hesap tablosu ve kelime işlemci programında kullanıcı için kolaylık sağlayan, işlemleri otomatik hâle getiren komutları araştırınız. Mesela: madde işaretleri ve numaralandırma gibi.

5. DÖNGÜLER



Programlamada bilgisayarın olabildiğince çok iş yapması ve bunları kısa zamanda bitirmesi istenir. Olabildiğince küçük ve az hataya sahip program yazarak, bu isteğimize ulaşmaya çalışırız. Bu sebeple defalarca yapılan işlemleri döngü halinde yazabiliriz.

"**Döngü** – For, **İken** – While, **Git** – Goto" gibi komutlar ile döngü yapabiliriz.

5.1. Döngü

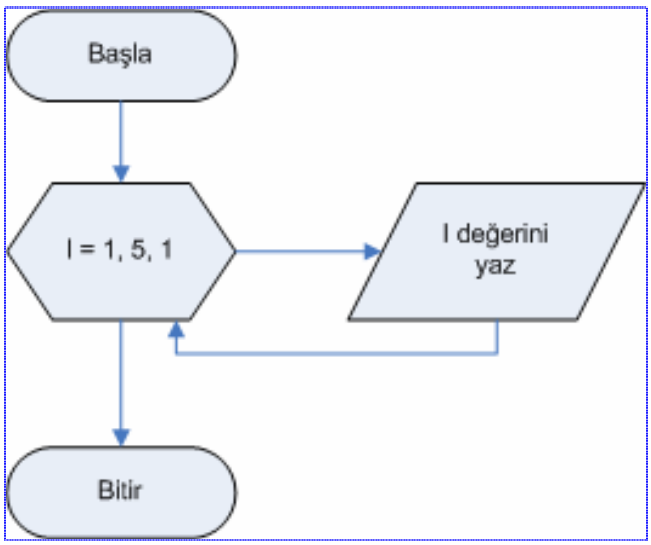
Ekrana 1'den 5'e kadar olan sayıları yazmak için, bu şekilde defalarca aynı satırı tekrar ederek yazabilirsiniz.

1. Başla
2. Yaz; 1
3. Yaz; 2
4. Yaz; 3
5. Yaz; 4
6. Yaz; 5
7. Bitti

Aynı programı bir de "**Döngü**" döngüsü ile daha kısa program olarak yapalım:

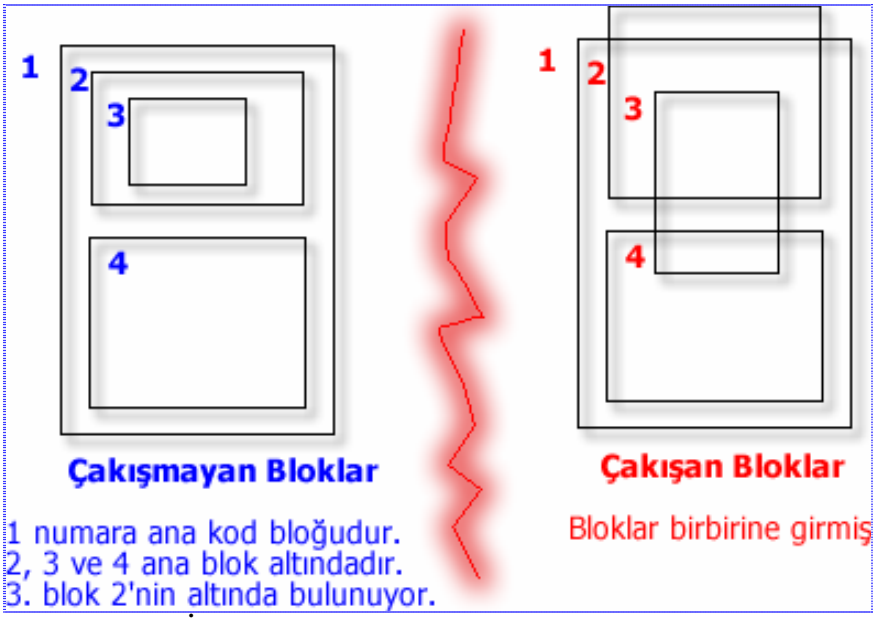
1. Başla
2. Sayısal I
3. Döngü I = 1, 5, 1
4. Yaz; I
5. Döngü Bitti
6. Bitir

Döngü sınırını tek hamlede daha da büyütebilir veya azaltabilirsiniz. Fazla kod yazmadan çok iş yapabilirsiniz. Döngünün ne zaman biteceğini de bir şarta bağlamanız gerekir. Döngü içi çok uzun bir program ise, açıklama satırları ile döngünün ne iş yaptığını belli yerlere yazabilirsiniz.



? Burada 1'den 5'e kadar olan sayılar ekrana yazılır. Siz de klavyeden girilen "N" sayısına kadar ekrana sayıları listeleyiniz.

İç içe döngü yapabilirsiniz. Ama döngülerin blok halinde yazımında birbirleri ile kesişmemelerine dikkat etmek gereklidir. Yani içteki döngünün başı ve sonu dıştaki döngünün başı ve sonu ile karışmamalıdır. Uzun programlarda **açıklama satırları** ile blokları belirterek birbirinden ayırt edilmesinde yardımcı olabilirsiniz.



Resim 5.1: İç içe blok yaparken başlangıç ve bitişlere dikkat ediniz

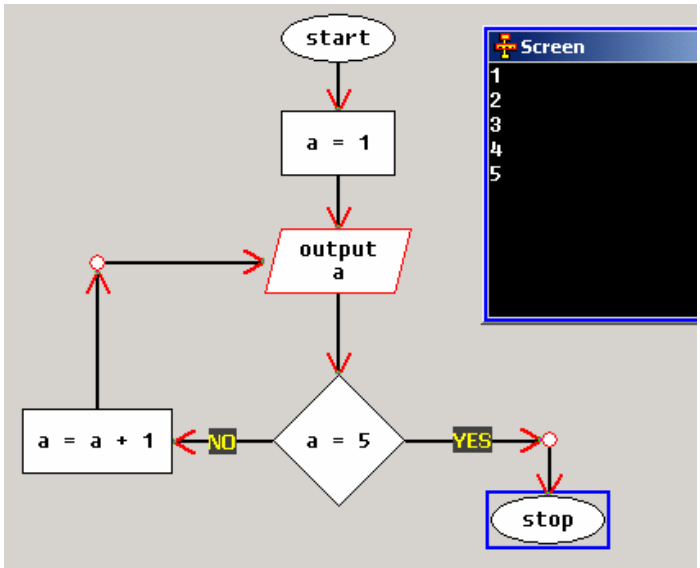
5.2. Şartlı Döngü

Belli bir şarta kadar dönen döngülere "**şartlı döngüler**" denir. Hatalı tasarlanırsa belki hiç çalışmayabilir, belki de döngüden hiç çıkılamayabilir.

1. Başla
2. Sayısal I
3. I = 1
4. İken (I < 5)
5. Yaz; I
6. I = I + 1
7. İken Bitti
8. Bitir

Bitmeyen döngülere "**sonsuz döngü**" diyoruz. Program hiç bitmeyeceği için ekrandaki her şey donup kalır. Kilitlenen bir program nasıl durdurulabilir?

Programların yazımında, açıklama satırlarının olduğu yerlerde bir problem var ise, ya da onlardan birini yazmayı unutursanız, sonsuz döngü elde edersiniz.



Resim 5.2: Bir döngünün akış şeması

❓ "Döngü (For) ve İken (While)" komutları ile aynı işi yapan programı yapabilirsiniz. Farklarını arkadaşlarınız ile tartışınız.

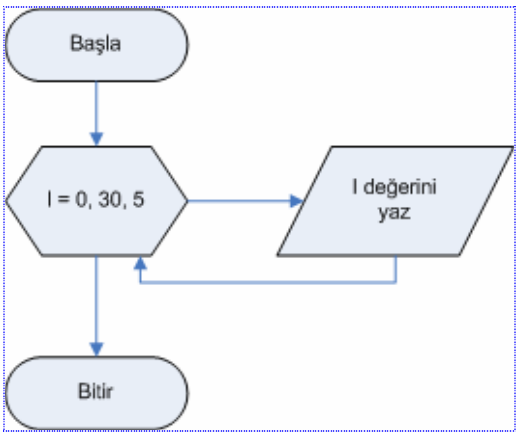
Örnek:

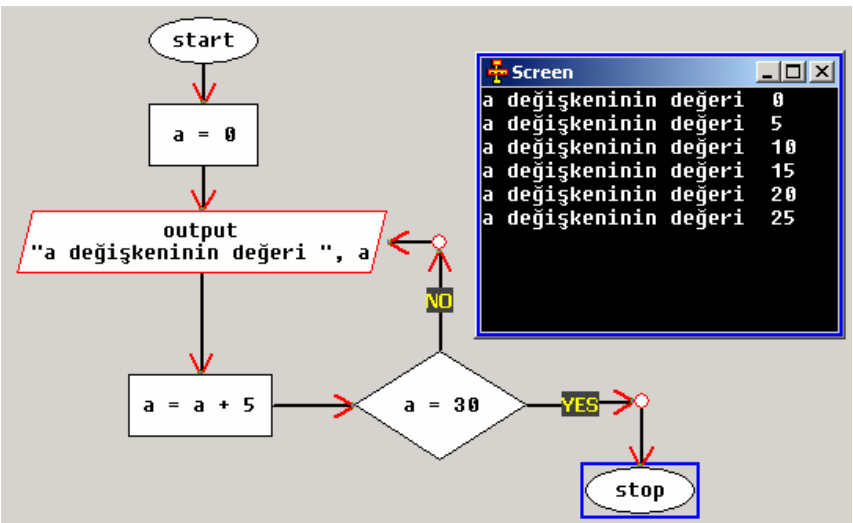
```
Başla //30 öğrenci notu girilir, ortalaması hesaplanır
      Sayısal öğrenciNotu, Toplam, sayaç, Ortalama
      öğrenciNotu = 0
      Toplam = 0
      sayaç = 0
      Ortalama = 0
      İken (sayaç < 30)
                Oku; "Öğrenci notunu giriniz", öğrenciNotu
                sayaç = sayaç + 1  //Döngü sayacı ilerletiliyor
                Toplam = Toplam + öğrenciNotu
      İken Bitti
      Ortalama = Toplam / sayaç
      Yaz; "Öğrenci notlarının ortalaması ", Ortalama
Bitir
```

Sıra No	öğrenciNotu	Toplam	sayaç	Ortalama
1	55	55	1	0
2	25	80	2	0
3	87	167	3	0
4	3	170	4	0
…				
30	82	2150	30	71.7

5.3. Adımlı Döngü

"Döngü - **For**" döngüsünde başlangıç ve bitiş değerlerini belirttikten sonra, gerekirse Basic dilindeki "**Step**" komutu gibi artım değerini değiştirebiliriz. Eğer ilerleme rakamını negatif olarak yazarsanız geri sayan döngünüz olur. Artım ve eksiltme işlemlerinde döngü başlangıç ve bitiş değerlerine dikkat ediniz. Yoksa ya döngü bitmez ya da hiç döngü çalışmayabilir.





Resim 5.3: Adımlı olarak döngü yapmak

1. Başla
2. Sayısal I
3. Döngü I = 0, 30, 5
4. Yaz; "I değişkeninin değeri " & I
5. Döngü Bitti
6. Bitir

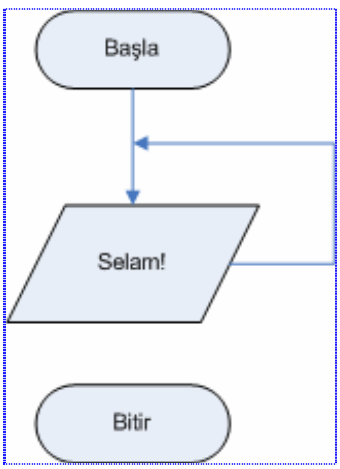
5.4. “Git” Komutu

“**Git** veya Goto” komutu, birçok programcı tarafından elden geldiğince az kullanılması gerektiğine inanılan bir komuttur. “Git” deyimi gerçekten de, çok yoğun kullanıldığında programı takip etmek neredeyse imkânsız hale gelir. Bu tür programlara “**spagetti programlar**” da denir.

“Git” ile istediğiniz program kısmına dallanılabilir. Akış çok rahatlıkla istenen noktadan devam ettirilebilir. Bu serbestlik programın okunaklılığını azaltır. Yapısal programlamaya ters bir durumdur. Çok gerekmedikçe, bu komutu kullanmamaya çalışınız. Eğer işin içinden bir türlü çıkamıyorsanız “Git” komutunu kullanabilirsiniz.

```
Başla
      Etiket Döngü
      Döngü:
         Yaz; "Selam"
         Git Döngü
Bitir
```

1. Başla
2. Yaz “Selam!”
3. Git 2
4. Bitir



Resim 5.4: Sonsuz döngü yapma örneği

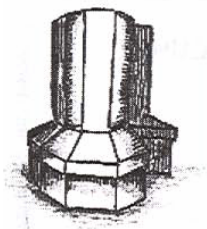
Aşağıdaki soruların akış şemalarını çiziniz ve test değerleri ile deneyiniz:

1. Klavyeden girilen 5 adet not bilgisinin ortalamasını alan programı yapınız.
2. Klavyeden girilen 10 adet notun en büyük ve en küçüğünü bulan programı yapınız.
3. Klavyeden girilen 5 adet sayının 10’dan büyük olanlarını sayan programı yapınız.
4. Klavyeden 0 sayısı girilene kadar sayılar okutunuz. Girilen sayıların 2 katını alarak ekrana sonucu yazdırınız.

5. 30 kişilik sınıfta, yaşı 13 ile 15 arasında olanların sayısını bulan programı yapınız.
6. 30 kişilik sınıfta yaşı 13, 14, 15 ve 16 olanların sayısını ayrı ayrı bulan programı yapınız.
7. Klavyeden girilen 5 adet sayının tek tek karelerini alan programı yapınız.
8. Klavyeden 3 not girilir. İlk notun %30, ikinci notun %30 ve son notun da %40'ını bulan programı yapınız. Sonuç olarak da 3 notun yüzdelerini toplayıp ekrana yazdırınız.
9. Klavyeden bir tam sayı okutunuz. Bu sayı ile klavyeden okunan diğer 10 sayıyı çarpma işlemi uygulayınız, sonuçları ekrana yazınız.
10. Klavyeden girilen 10 sayıdan 5'ten büyük olanların yarısını, 5'e eşit ve küçük olan sayıların 2 katını bulan programı yapınız.
11. Bir komisyoncu sattığı mallardan fiyatı 50 YTL kadar olanlardan %3, daha fazla olanlardan ise %2 komisyon almaktadır. Klavyeden girilen 5 malın komisyonlarını bularak, toplam komisyonu hesaplayınız.
12. Klavyeden 5 adet yarıçapı verilen çemberlerin alanını ve çevresini hesaplayan programı yapınız.
13. Klavyeden girilecek N sayısı kadar nottan en büyük ve en küçük olanı bulan programı yapınız.
14. İç içe döngüler ile saat: dakika: saniye olarak saat yapınız. Saat 0 ile 23, dakika 0 ile 59 ve saniye de 0 ile 59 arasında ilerleyecektir.
15. Klavyeden girilen 100'lük sistemdeki 5 notu; 0, 1, 2, 3, 4 ve 5 olacak şekilde ekrana yazan programı yapınız.
16. Sayısal olarak girilen bir ay bilgisini ekrana "Ocak, Şubat, Mart veya diğer aylardan biri…" şeklinde yazan programı yapınız.
17. Haftanın günü (Pazartesi, Salı, …) girilince, o günün haftanın kaçıncı günü olduğunu bulan programı yapınız.
18. Fiyat ve KDV oranı ayrı ayrı girilen 5 malın toplam fiyatını hesaplayınız.
19. Klavyeden dakika olarak girilen 5 şarkının toplam süresini saat olarak hesaplayan programı yapınız.
20. Girilen işlem türüne (* / - +) göre iki sayıyı işleme alıp sonucunu ekrana yazan programı yapınız.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Belli bir şarta bağlı olarak biten, sık yapılan işlemlerin döngü halinde programın akış şemasını yapınız.	➢ Her şey döngü hâlinde olmak zorunda değildir. Tekrar eden işlemleri döngü hâline getiriniz. ➢ Mesela, ekrana 3'er 3'er, 1'den 20'ye kadar sayıları yazdırınız.
2. Döngünün bitmesini beklemeden belli bir şarta ulaşınca döngüden çıkınız.	➢ Önceki örnekte; mesela döngü değişkeni 13 olunca döngünün bitmesini sağlayınız.
3. "Şartlı döngü"ye başlamadan önce şartın ilk değerini belirleyiniz.	➢ Önceki örneği "şartlı döngü" hâline getiriniz.
4. "Şartlı döngü" içinde artım ve eksilme komutları ile ilerleyiniz.	



Cray'in yeni modelini duydunuz mu?

O kadar hızlıymış ki sonsuz döngüleri bile 6 saniyede bitiriyormuş.

ÖLÇME VE DEĞERLENDİRME

OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız.

1. Döngü komutları sayesinde programın akışını değiştirebiliriz. ( )

2. Döngünün başlangıç, artım ve bitiş değerlerini belirlememiz gerekir. ( )

3. Hiç bitmeyen döngülere sonsuz döngü denir. ( )

4. Döngü içinde döngü tanımlanamaz. ( )

5. Döngü ile yapılan programlar daha hızlı çalışır. ( )

6. "Git" komutu programın okunaklılığını artırır. ( )

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modül ile kazandığınız yeterliği, öğretmeniniz işlem basamaklarına göre 0 ile 7 puan arasında olacak şekilde değerlendirecektir.

DEĞERLENDİRME KRİTERLERİ	Puan
Veri girme komutu ile klavyeden bilgi okuma	
Veri çıkış komutu ile ekrana yazı yazdırma	
Bir dış kaynaktan (internet, tarayıcı, dijital fotoğraf makinesi) bilgi alma, yazıcıdan çıktı alıp, ses kartından müzik çalma	
Tanımlama yazımında ilk değeri verme, değişken türünü belirleme	
Değişkenler üzerinde matematiksel ve metin işlemleri yapma	
Gerekli yerlere açıklama satırları yaparak, programcıya bilgilendirme yapma	
Tanımlanan değişkenlere sayısal hesaplamalar yaparak değer aktarma	
Elde edilen değeri ekranda düzgünce gösterme	
Karmaşık matematiksel formüllerde gerekli yerlere parantez ekleyerek, işlem önceliklerini belirleme	
Metin birleştirme operatörü ile birden fazla metin değeri birleştirme, uygun fonksiyonlar ile metin içinde arama yapma	
Kullanıcıya bir soru sorarak, cevabına göre doğru yönlendirme, girilen veriyi sabit veya hesaplanan değişkenler ile karşılaştırma	
Belli bir şarta bağlı olarak biten, sık yapılan işlemlerin döngü hâlinde programını yazma	
Döngünün bitmesini beklemeden belli bir şarta ulaşılınca döngüden çıkma	
"Şartlı döngü" içinde artım ve eksilme komutları ile ilerleme	
Toplam (en fazla 98 puan olabilir)	

DEĞERLENDİRME

Yaptığınız değerlendirme sonucunda eksikleriniz varsa öğrenme faaliyetlerini tekrarlayınız.

Modülü tamamladınız, tebrik ederiz. Programlamanın çok temel bir modülü olduğu için belli zamanlarda bu modülü tekrar gözden geçiriniz.

Öğretmeniniz size çeşitli ölçme araçları uygulayacaktır. Öğretmeninizle iletişime geçiniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1 CEVAP ANAHTARI

1	Y
2	çıkış – output
3	Y
4	Y
5	Y
6	Y
7	D
8	D
9	D
10	A

ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI

1	D
2	Y
3	Y
4	Y
5	Y
6	D
7	Y (çevrimden sonra mümkündür)
8	Y
9	D (ana bellek ile sınırlıdır.)
10	Y
11	D
12	C (VB' de problem olmaz)

ÖĞRENME FAALİYETİ-3 CEVAP ANAHTARI

1	D
2	Y
3	D
4	D
5	D
6	Y
7	B
8	B
9	0.524
10	D

ÖĞRENME FAALİYETİ-4 CEVAP ANAHTARI

1	D
2	2
3	5
4	2

ÖĞRENME FAALİYETİ-5 CEVAP ANAHTARI

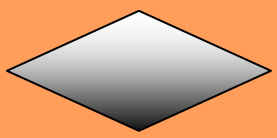
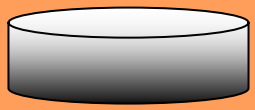
1	Y
2	D
3	D
4	Y
5	Y
6	Y

Cevaplarınızı cevap anahtarları ile karşılaştırarak kendinizi değerlendiriniz.

SÖZLÜK

İsim	Okunuş	Anlam
debug	dibag	Windows'un böcek ayıklama programı
default	difolt	varsayılan
Delphi	delfi / delfay	Yunanistan'da Parnassus Dağı eteklerinde eski bir şehir
demonstration	demonstreyşın	reklam, tanıtım, demo
design	dizayn	tasarım, dizayn
destination	destineyşın	hedef
device	divays	alet, aygıt
diagnostic	dayagnostik	sebebi bilinmeyen bir hatayı giderme, diyagnoz
dialog	dayalog	diyalog, söyleşme
dimension	daymenşın	boyut
disassemble	disessembl	parçalamak
dot	dot	nokta, ekranın fiziksel en küçük ışık noktası (dpi *di pi ay* – dots per inch)
draft	draft	taslak
driver	drayvır	sürücü
exceed	eksiid	aşmak
entry	entri	giriş (entrance)
environment	envayırmınt	ortam, çevre
evaluation	evalueyşın	değerlendirme
expanded	ekspendid	genişletilmiş
extension	ekstenşın	uzantı
fail	feyl	hata, failure – başarısızlık
false	fols	mantıksal yanlış
FAQ	ef ey kyu	Frequently Asked Questions – Sık Sorulan Sorular
feature	fyuçır	özellik, aksam
file	fayl	dosya
float	flout	kayar noktalı gerçel sayılar
frame	freym	çerçeve, iskelet, kare
function	fankşın	görev, işlev, fonksiyon
GUI	ci yu ay	Graphical User Interface – grafiksel kullanıcı arabirimi
guide	gayd	önder, kılavuz
handle	hendl	tutamaç, kulp, tanıtıcı
hexadecimal	heksadesimıl	onaltılık sayı sistemi
HTML	eyç ti em el	Web sayfalarının temelini oluşturan dil
icon	aykın	simge
identifier	aydentifayır	tanımlayıcı
inch	inç	2.54 santimetre
information	infomeyşın	bilgi, enformasyon
initiation	inişieyşın	ilkleme, kullanıcı ayarlarının saklandığı dosyalar (*.ini)

AKIŞ ŞEMASI SEMBOLLERİ

Şekil	Görev
	Giriş veya çıkış (*input* ve *output*) Kullanıcı etkileşimi için gereklidirler
	İşlem, hesaplamalar (*process*, *calculation*) Bilgisayarın asıl yaptığı işler burada gerçekleşir
	Karar, eğer (*decision*, *if*) Program iki ihtimale göre akışını değiştirir, her ihtimalin ne işlem yapacağını göstermeyi unutmayınız
	Alt program, fonksiyon (*sub program*, *function*) Akış diyagramı karmaşık hale gelince, tekrarlanan belli kısımlarını alt program haline getirebilirsiniz
	Akış çizgisi (*flowline*) Genellikle yukarı ve sola doğru çizilmezler
	Başla veya Bitir (*Start* ve *Stop*) Her programın bir başlangıç ve bir bitişi vardır
	Birleştirici, bağlantı noktası (*connector*) Birçok sayfadan oluşan akış şemalarını birbirine bağlar
	Döngü (*for*, *cycle*) Başlangıç, artım ve sonlanma şartlarını iyi belirleyiniz
	Dosya (*file*, *storage*) Saklanması gereken bilgileri kaydederiz

KOD ÖRNEKLERİ

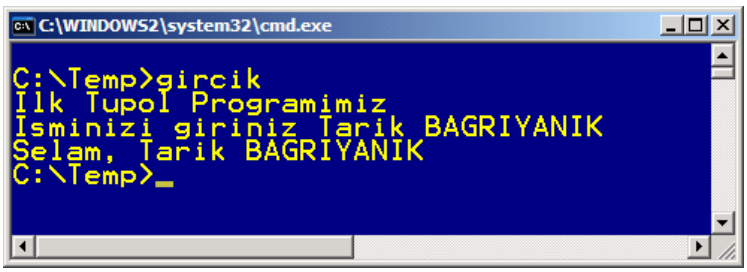
Not: Her dilin kendine göre avantajı bulunmaktadır. Verilen örnekleri yaptığınızda o dilleri öğrenmiş olmayacaksınız. Asıl amaç, anlatılmak istenen konunun uygulanmasıdır, somut ve anlaşılır hale gelmesidir.

Tupol dilinde giriş çıkış örneği

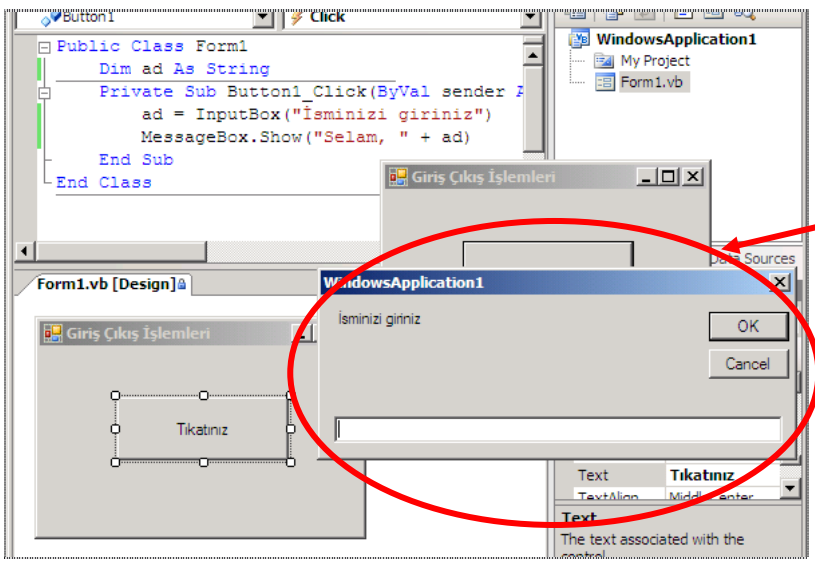
```
PROGRAM GirisCikisProgrami;

   TANIM
           Harf Ad[30];
   TANIMSONU

Basla
         Yazi(#i,"Ilk Tupol Programimiz",\n);
         Yazi(#i,"Isminizi giriniz ");
         Oku(#c,Ad);
         Yazi(#i,"Selam, ");
         Yazi(#c,Ad);
Bitti.
```



Tupol'de giriş çıkış programımızın çalışma anı



Basit bir veri giriş diyalog kutusu

VB.NET ile yapılan giriş çıkış programımızın çalışma anı

Visual Basic dilinde giriş çıkış örneği

```
Dim ad As String
ad = InputBox("İsminizi giriniz")
MessageBox.Show("Selam, " + ad)
```

VB.NET'de veri girmek için "**InputBox**" komutu, basit mesaj göstermek için ise "**MessageBox**" komutu kullanılabilmektedir.

Not: "Option Explicit" seçeneği aktif olduğu için VB.NET'de de değişken tanımlamak zorunda kalınmıştır.

C dilinde KDV hesaplama örneği

```
main()
{
          float Fiyat, Sonuc;
          printf ("Fiyat giriniz");
          scanf ("%f", &Fiyat);
          Sonuc = Fiyat * 1.18;
          printf ("Sonuc %8f", Sonuc);
}
```

Basic dilinde karar deyimi örneği 1

```
  IF (4 < 6) THEN
   PRINT "4 6'dan küçüktür."
  ELSE
   PRINT "4 6'dan büyüktür." 'sizce bu satır hiç çalışır mı?
  END IF
END
```

Basic dilinde karar deyimi örneği 2

```
  sonuc = (4 < 6) 'şart sonucu değer olarak nedir?
  IF sonuc THEN
   PRINT "4 6'dan küçüktür."
  ELSE
   PRINT "4 6'dan büyüktür."
  END IF
END
```

Basic dilinde karar deyimi örneği 3

```
  INPUT "Ne kadar ücret alıyorsunuz?"; ucret
  INPUT "Almak istediğiniz elbisenin fiyatı nedir?"; fiyat
  IF (ucret > 1000) THEN
   IF (fiyat < ucret) THEN
    PRINT " Elbiseyi almak için yeterli ücretiniz var."
   END IF
  END IF
END
```

Basic dilinde karar deyimi örneği 4

```
  INPUT "Ne kadar ücret alıyorsunuz?"; ucret
  INPUT "Almak istediğiniz elbisenin fiyatı nedir?"; fiyat
  IF (ucret > 1000) AND (fiyat < ucret) THEN
    PRINT "Almak için yeterli ücretiniz var."
  END IF
END
```

Basic dilinde "Durum" kullanımı 1

```
INPUT "Notunuzu giriniz "; notum
SELECT CASE notum
CASE IS > 84
  PRINT "Çok iyi!"
CASE IS > 69
  PRINT "İyi"
CASE IS > 44
  PRINT "Orta"
CASE IS > 24
  PRINT "Zayıf"
CASE ELSE
  PRINT "Çok kötü!"
END SELECT
END
```

Basic dilinde "Durum" kullanımı 2

```
SELECT CASE
CASE (cevap < 10)
PRINT "10'dan küçük"  '10'dan küçük sayılar
CASE (cevap < 12)
PRINT "12'den küçük"  '12'den küçük sayılar!
END SELECT
```

C ve Java gibi dillerde "Durum" komutunda dikkat etmemiz gereken bir konu vardır. Her durum bloğunun sonuna "break;" komutu yazılmalıdır. Yazmayı unutursanız, ilk çalışan şarttan itibaren tüm alttaki durumların sonuçlarını da çalıştırır. En son ihtimalde ise "break;" yazmanıza gerek yoktur. Zaten program kaldığı yerden devam eder.

C dilinde "Durum" kullanımı

```
#include <stdio.h>
main ()
{
char tus;
printf ("Bir tuşa basınız");
scanf ("%c", &tus);
 switch (tus) {
  case 'a': printf ("a tuşuna bastınız\n"); break;
  case 'b': printf ("b tuşuna bastınız\n");  //break; gerekmez…
 }
}
```

Bir dilden başka bir dile kodları taşırken hedef dilin kurallarını iyi bilmeniz gereklidir. Programcılar genelde önce basit bir dilde programlarının prototiplerini hazırlar, daha sonra asıl dile bu kodu aktarırlar ve kodu iyileştirirler.

Basic dilinde döngü örneği 1

```
 FOR I = 1 TO 5
  PRINT I
 NEXT I
END
```

Basic dilinde döngü örneği 2

```
I = 1          'başlangıç değeri
 WHILE I < 5   'bitme şartı
  PRINT I
  I = I + 1    'artım değeri
 WEND
END
```

Döngünün bitme şartı dışında başka ihtimallerde de bitmesi gerekli ise "çıkış - **exit**" komutu kullanılır:

Basic dilinde döngü örneği 3

```
I = 1                          'başlangıç değeri
 DO
  PRINT I
  I = I + 1                    'artım değeri
  IF I=6 THEN EXIT DO        'I'nın 8 olmasını beklemiyoruz
 LOOP WHILE I < 8              'bitme şartı
END
```

Basic dilinde döngü örneği 4

```
 FOR a = 1 TO 30 STEP 5 '5'er 5'er gidiyor
  PRINT "a değişkeninin değeri ";a
 NEXT a
END
```

Basic dilinde döngü örneği 5

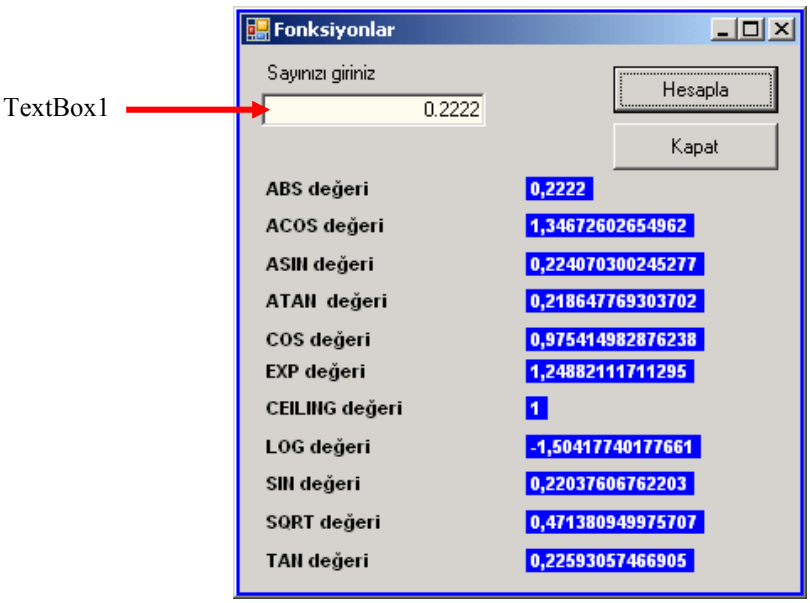
```
 FOR I = 80 TO 50 STEP -5 '5'er 5'er geri gidiyor
  PRINT "I değişkeninin değeri ";I
 NEXT I
END
```

C dilinde döngü örneği

```
main ()
{
int i;
 for (i = 1; i <= 5; i++) {
  printf ("%d\n", i);
 }
}
```

Basic dilinde "Goto" örneği

```
 Basla:
  PRINT "Selam!"   'sonsuz selam size
 GOTO Basla
END
```



Matematik komutları örneği

Visual Basic dilinde sayı fonksiyonları örneği

```
Label12.Text = Math.Abs(Val(TextBox1.Text))
Label13.Text = Math.Acos(Val(TextBox1.Text))
Label15.Text = Math.Asin(Val(TextBox1.Text))
Label17.Text = Math.Atan(Val(TextBox1.Text))
Label19.Text = Math.Cos(Val(TextBox1.Text))
Label14.Text = Math.Exp(Val(TextBox1.Text))
Label16.Text = Math.Ceiling(Val(TextBox1.Text))
Label18.Text = Math.Log(Val(TextBox1.Text))
Label20.Text = Math.Sin(Val(TextBox1.Text))
Label21.Text = Math.Sqrt(Val(TextBox1.Text))
Label23.Text = Math.Tan(Val(TextBox1.Text))
```

Visual Basic dilinde metin fonksiyonları örneği

```
Label12.Text = UCase(TextBox1.Text)
Label13.Text = LCase(TextBox1.Text)
Label15.Text = Len(TextBox1.Text)
Label17.Text = Trim(TextBox1.Text)
Label19.Text = (TextBox1.Text).Substring(0, 4)
Label14.Text = (TextBox1.Text).Substring(Len(    _
            TextBox1.Text) - 4)
Label16.Text = (TextBox1.Text).Substring(2, 4)
Label18.Text = Replace(TextBox1.Text, "e", "ü")
Label20.Text = Val(TextBox1.Text)
```

Metin komutları örneği

İyi bir programcı, bilgilerini paylaşır

ÖNERİLEN KAYNAKLAR

- en.wikipedia.org/wiki/Touchscreen
- www.atmel.com/products/8051
- www.c-jump.com
- www.ceebot.com
- www.chessopolis.com
- www.cs.uga.edu/~william/csci1301/keyboard.html
- www.danbbs.dk/~erikoest/ascii0.htm
- www.emteachline.com
- www.edge.org/digerati/simonyi/simonyi_p1.html
- www.informit.com/articles/article.asp?p=30110&seqNum=9&rl=1
- www.microchip.com
- www.labcenter.co.uk
- www.ondotnet.com/pub/a/dotnet/2001/07/30/vb7.html?page=2
- www.onelife.com/evolve/brain.html
- www.sharpdevelop.net
- www.sourgeforge.net
- www.teacherschoice.com.au/software.htm
- www.transhumanist.com/volume1/moravec.htm
- www.wikipedia.org
- www.winguides.com/scripting
- www.yunus.projesi.com

KAYNAKÇA

- AYFER Can Uğur, **Kim Güler Bilgisayarlara?**, Pusula Yayınevi, İstanbul, 1998
- BAĞRIYANIK Tarık, **Programlama Ders Notları ve Uygulamalı Genel Programlama Kitabı** (www.yunus.projesi.com)
- SANKUR Bülent, **Bilişim Sözlüğü**, Pusula Yayıncılık, 2004 (www.bilisimsozlugu.com)
- NIIT Global Learning Solutions, Fundamentals of Programming
- VATANSEVER Fahri, **Algoritma Geliştirme ve Programlamaya Giriş**, Şeçkin Yayınevi, Ankara, 2004
- WALLACE Wang, **Beginning Programming for Dummies**, Wiley Basımevi, Indianapolis, 2004