İNTERNET DESTEKLİ PROGRAMLAMA MANTIĞI ÖĞRETİMİ

Taner ARABACIOĞLU

YÜKSEK LİSANS TEZİ
BİLGİSAYAR EĞİTİMİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2006
ANKARA

İNTERNET DESTEKLİ PROGRAMLAMA MANTIĞI ÖĞRETİMİ

Taner ARABACIOĞLU

YÜKSEK LİSANS TEZİ
BİLGİSAYAR EĞİTİMİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2006
ANKARA

Taner ARABACIOĞLU tarafından hazırlanan İNTERNET DESTEKLİ PROGRAMLAMA MANTIĞI ÖĞRETİMİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Halil İbrahim BÜLBÜL
Tez Yöneticisi

Bu çalışma, jürimiz tarafından Bilgisayar Eğitimi Anabilim Dalında Yüksek lisans tezi olarak kabul edilmiştir.

Başkan: : Prof. Dr. Ali Paşa AYDIN

Üye : Yrd. Doç. Dr. Halil İbrahim BÜLBÜL

Üye : Yrd. Doç. Dr. Hasan Hüseyin SAYAN

Tarih : ...../……/……

Bu tez, Gazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Taner ARABACIOĞLU

İNTERNET DESTEKLİ PROGRAMLAMA MANTIĞI ÖĞRETİMİ
(Yüksek Lisans Tezi)

Taner ARABACIOĞLU

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Temmuz 2006

ÖZET

Programlama, herhangi bir problemin bir programlama dili kullanılarak çözülmesi için yazılan kod satırlarına verilen isimdir. Programlama mantığı öğretimi ise, programlama öğretiminin ilk ve en önemli basamağıdır. Her bilim dalında olduğu gibi başlangıcın olumlu olması diğer bir deyişle temelin sağlam atılması konunun anlaşılmasında oldukça önem taşımaktadır. Programlama dili eğitimi alan öğrencilerin yabancı dile ve programlamaya karşı olan olumsuz önyargıları, programlama dilinin öğretimini zorlaştırmaktadır. Eğitim kurumlarında programlama mantığı genelde teorik bir yöntemle verilmektedir. Teorik yöntemin öğretimde hem sıkıcı hem de çok etkili bir yöntem olmadığı bilinmektedir. Öğretimin uygulamalı bir yöntemle yapılması teorik yönteme göre başarıyı arttırmaktadır. Bu çalışma ile programlama mantığı öğretiminde kullanılmak üzere, bir uygulama dili tasarlanmıştır. Uygulama dili kullanılarak programlama mantığı öğretiminde, programlama öğretimi etkinliğini arttırmak için teorik bilgilerin uygulamaya dönüştürülebilmesi amaçlanmıştır.

Bilim Kodu : **702.6.004**
Anahtar Kelimeler : **Programlama mantığı, uygulama dili**
Sayfa Adedi : **43**
Tez Yöneticisi : **Yrd. Doç. Dr. Halil İbrahim BÜLBÜL**

INTERNET SUPPORTED PROGRAMMİNG LOGIC INSTRUCTION

(M.Sc. Thesis)

Taner ARABACIOĞLU

GAZI UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

July 2006

ABSTRACT

Programming is the name given for the code lines written for solving any problem by using a programming language. Teaching of programming logic is the first and most important phase of programming teaching. Like in each science branch, having the starting point as positive, that is to say having a sound ground is very important for understanding of the subject. The negative prejudgements of the students having programming training against the foreign language and programming make teaching of a programming language difficult. Programming logic in schools are generally given by means of a theoretical method. It is known that theoretical method is boring and not very effective one. Giving the teaching through an applied method resluts in higher success than theoretical method. In this study, an application language is designed to use in the teaching of algoritm. Using the theoritical knowledge in application is aimed to increase the activities of programming teaching, using the application language in the teaching of algoritm.

Science Code : 702.6.004
Key Words : Programming Logic, application language
Page Number: 43
Adviser : Assistant Prof. Dr. Halil İbrahim BÜLBÜL

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren Sayın Hocam Yrd. Doç. Dr. H. İbrahim BÜLBÜL ’e teşekkürü bir borç bilirim. Ayrıca Adnan Menderes Üniversitesi Fen Edebiyat Fakültesi Matematik Bölümü öğretim üyesi Sayın Yrd. Doç. Dr. Ali FİLİZ’ e yardım ve katkıları için teşekkür ederim.

İÇİNDEKİLER

Sayfa

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

1. GİRİŞ

Bilişim teknolojilerinin hızla ilerlemesi sonucu günlük hayatta yapılan işlerin çoğu otomasyona geçmiş durumda veya geçiş hazırlıkları hızla devam etmektedir. Rutin yapılan işler eskiye oranla bilişim teknolojileri sayesinde çok hızlı bir biçimde gerçekleştirilmektedir.

İnternet teknolojisinin ve kullanımının hızla gelişmesiyle, ülkeler hatta kıtalar arası mesafeler ortadan kalkmıştır. Dünya global bir köy haline gelmiştir. Bu sayede büyük yazılım şirketleri veya program ihtiyacı olan herhangi bir firma bir başka ülkedeki programcı ile ortak çalışabilmektedir. Bunun sonucu olarak da programcılar mekan kavramı olmaksızın nerede olursa olsun isteyen herkese hizmet verebilmekte ve kendi ülkesine katma değer sağlamaktadır. Bunun yanı sıra gelişmiş ülkelerde üretilen programlara ödenen lisans ücretleri de düşünülürse ülke ekonomisine yapılabilecek katkı oranı artmaktadır. Hatta ülke dışından alınan programların benzerlerinin yapılması ve ülke dışına satılması ekonomik kalkınma ivmesini artacaktır. Bilişim teknolojileri alanında yazılım üreten Hindistan, Çin, İrlanda gibi ülkeler son zamanların en iyi örnekleridir. Hindistan'da yazılım sektörünün bu denli gelişmiş olmasına etkenlerden biri de İngilizce' nin ülkede resmi dil olarak kullanılması denilebilir.

Ülkemizde ise eğitim-öğretim süresi boyunca öğrencilere bir türlü kazandırılamayan yabancı dil seviyesi ve bir türlü kırılamayan önyargılar, programlama öğretiminin önündeki en büyük engellerden bir kaçıdır. Piyasada kullanılan programlama dillerinin tümü İngilizce olduğu düşünüldüğünde konunun önemi daha da artmaktadır.

Programlama, herhangi bir problemin bir programlama dili kullanılarak çözülmesi için yazılan kod satırlarına verilen isimdir. Bilgisayarın sadece, programcının verdiği komutları yerine getirdiği düşünüldüğünde problemin iyi anlaşılması ve iyi analiz edilmesinin önemi ortaya çıkmaktadır. Programlama dillerine ait komutlar birbirleri

arasında farklılık göstermesine rağmen, çözüm için kullanılacak programlama mantığı tüm dillerde benzerdir.

Diğer bir deyişle algoritma; tüm bilgisayar programlarının tasarımı aşamasında yararlanılan ve işin tamamlanması için gerekli işlemlerin kendi dilimizde tarif edildiği bir belgedir. Programlama mantığı öğretiminde konuşma dilinin kullanılmasındaki amaç, her bir adımın tek bir anlam içerecek şekilde açıklayıcı olması ve herhangi bir programlama diline bağlı olmaksızın geliştirilmesindendir.

Algoritma tasarımı sırasında kullanılan komutlar kısa öz ve anlaşılır olmalıdır. Algoritma bir işin tarif edildiği işlem basamakları olduğundan, işlem sırası oldukça önemlidir. Bazı işlemlerin sırasında değişiklikler yapmak mümkün olabilse de yapılan bu değişiklik sonucu etkilememelidir.

Algoritma yazımında programcının bilgisayara yapmasını istediği işlemleri söylediği düşünülerek emir kipinde komutlar kullanılır. Örneğin iki sayının çarpılması gerekiyorsa; bu sayıları tarif eden değişken simgeleri de A ve B ise komut; "C = Çarp A, B" veya "A ile B' yi çarp sonucu C ye yükle" şeklinde olabilir. Bazı adımlarda ifade kullanmak yerine fonksiyonun kendisini de yazılabilir. Bu durum için üstteki örnek tekrar; "C=A*B" şeklinde ifade edilebilir.

Yapılan bu çalışma ile algoritma mantığının kolay bir biçimde öğretilmesi amaçlanmaktadır. Algoritma mantığının, bilgisayar programcısı olacak kişilere kolayca kavratılabilmesi açısından alışılandan farklı bir yöntemle sunulmaktadır. Programlama mantığı öğretiminin kolaylaştırılması ve basit bir arayüz kullanılarak öğrenciye bir editör alışkanlığı kazandırılması hedeflenmektedir.

2. KONUYLA İLGİLİ ÇALIŞMALAR

Bu Bölümde konu ile ilgili yapılmış olan çalışmalara yer verilmiş. Konu ile ilgili kaynaklar YÖK Dokümantasyon Merkezi, Gazi Üniversitesi, Adnan Menderes Üniversitesi Kütüphanelerinde temin edilen basılı kaynaklar ve internet yolu ile elde edilen çevrim içi kaynaklardan faydalanılmıştır.

Elde edilen kaynaklar tasnif edilmiş ve konu ile ilgili yönleri buraya aktarılmıştır.

Eker (2004), Algoritmayı Anlamak adlı çalışmasında, bir problemi çözebilmek için, gerekli olan, çözüm basamaklarını aşağıdaki gibi sıralamıştır:

1. Yöntem Geliştirme: Bir problemin tanımı yapıldıktan sonra çözüm için yöntem geliştirilmelidir. Böylece çözümün nasıl gerçekleştirileceği de belirlenir.

2. Girdi ve Çıktı Belirleme: Bir problemin çözümünde kullanılacak hammaddenin ve bu hammaddenin sisteme nasıl girileceği, üretilen sonuçların sistemden dışarıya ne şekilde sunulacağı belirlenir.

3. Çözümü Kağıt Üzerinde Gösterme: Çözümün daha iyi anlaşılması için geliştirme aşamasında, kağıt üzerinde şemalar haline getirilir. Böylece çözüm basamaklarının birbirleriyle ilişkileri ve bilgi akışı daha kolay izlenebilir.

4. Çözümü Deneme: Çözüm herhangi bir derleyicide uygulanır yani hayata geçirilir. Hatalar tespit edilir. Varsa gerekli düzeltmeler yapılır.

5. Çözümü Geliştirme: Çözüm, uygulama sırasında tespit edilen hatalardan arındırıldıktan sonra gelen eleştiriler doğrultusunda yada zaman içerinde ortaya çıkan ihtiyaçlar doğrultusunda geliştirilmesi beklenir. Aslında bu süreç tüm problem çözme sürecinin tekrarıdır.

Çözümü kağıt üzerinde gösterme aşamasında, akış diyagramları kullanılmaktadır. Bununla birlikte, derleyici komutlarının bazılarının Türkçe karşılıkları da algoritma öğretiminde kullanılmaktadır[1].

Algoritma ile ilgili olarak üstte ortaya konan çözüm basamakları, yapılan bu çalışmanın da temelini oluşturmaktadır. 4. basamak olan Çözümü Deneme ve 5. basamak olan Çözümü Geliştirme adımlarının programlama mantığı öğretimi sırasında uygulanamaması, çalışmanın çıkış noktasını oluşturmaktadır.

Sefer Kurnaz tarafından 2000 yılında Veri Yapıları ve Algoritma Temelleri isimli diğer bir çalışmada ise çalışmasında, algoritmanın belirli bir görevi yerine getiren sonlu sayıdaki işlemler dizisi olduğunu ve her algoritmanın sağlaması gereken şartları aşağıdaki gibi izah edilmektedir;

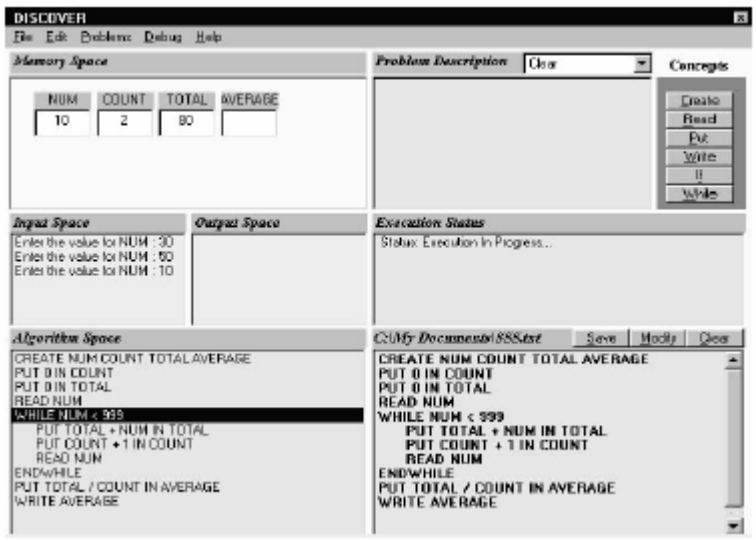
Girdi: Sıfır veya daha fazla değer dışarıdan verilmeli.
Çıktı: En azından bir değer üretilmeli.
Açıklık: Her işlem (komut) açık olmalı ve farklı anlamlar içermemeli.
Sonluluk: Her türlü olasılık için algoritma sonlu adımda bitmeli.
Etkinlik: Her komut kişinin kalem ve kağıt ile yürütebileceği kadar basit olmalıdır[2].

Yapılan bu çalışmada, bir programlama mantığı uygulama dili geliştirildiği düşünüldüğünde, tasarım ilkeleri olarak yukarıda belirtilen ilkeler dikkate alınmıştır.

Konuyla ilgili olarak, H.A.Ramadhan, Programming By Discovery adlı çalışmasında ise, yeni başlayanlar için programlama dili, az sayıda genel kavram, basit ve mantıksal yazım kuralları ve anlaşılması kolay az sayıdaki temel komutları içermelidir. Bu komutlar kullanıcının kısa ve basit programlar yazarak, problemleri çözebileceği şekilde olmalı ve kullanıcılar komut isimlerinden anlam çıkarabilmelidir. Örnek olarak assembly dilinde kullanılan LOAD ve STORE komutları gösterilebilir. Bilgisayar programlamayı öğretirken kullanılan dil, alt program veya özyineleme gibi kavramları kapsamadan az sayıdaki komut ile yapısal programlama deneyimi kazandırmalıdır. Kullanıcılar, programlamaya giriş

aşamasında bir uygulama dili kullanırlarsa, problem çözümünü daha kolay yapabilirler ve bilgilerini diğer programlama dillerine daha kolay aktarabilirler. Daha basit ve daha az kavram içeren dilleri öğrenmek, kullanıcı üzerindeki zihinsel yükü azaltır, programı derlemede ve anlamada yardımcı olur. Discover adlı sistemde kullanılan programlama dili, basit bir uygulama dilidir.

Discover sisteminde, fonksiyonlar, prosedürler, özyineleme ve diziler olmadığı için, kullanıcının dikkati temel programlama kavramlarına odaklamakta ve öğrenmeyi kolaylaştırmaktadır. Dil sadece create, put, read in, write out, while-end-while ve if-istrue-isfalse komutlarını içermektedir. Discover sisteminin çalışması aşağıdaki şekilde gösterilmiştir. Şekil 2.1'de Discover sisteminin bir görüntüsü bulunmaktadır[3].



Şekil 2.1 Discover sistemi

Üstte de görüldüğü gibi Discover sistemi, değişken içeriği, kavram, girdi, çıktı, çalışma durumu ve algoritma pencerelerinden oluşmaktadır. Yapılan çalışmaya oldukça benzemekte ancak, kullanıcı ile etkileşim yöntemi ve dilinin İngilizce olmasıyla tasarlanan uygulama dilinden farklılıklar göstermektedir.

Shackelford ve LeBlanc, Introducing Computer Science Fundamentals Before Programming adlı çalışmalarında ise, programlama dillerinin her bir kuşağı, yorumlamada artan istekleri ve temel algoritmik prensipleri uygulamada bir öncekine göre daha da karmaşıklaştığını belirtmişlerdir. Georgia Teknoloji Enstitüsü, bilgisayar bilimlerine yeni başlayan öğrenciler için daha önce kullanılan Programlamaya Giriş I ve Programlamaya Giriş II dersleri yerine Hesaba Giriş (Introduction to computing) ve Programlamaya Giriş (Introduction to Programming) derslerini vermektedir. Hesaba Giriş dersinde, hesap ve algoritmik prensiplerin temelleri verilmekte, Programlamaya Giriş dersinde ise, bir programlama dili kullanılarak, öğrencilere problem çözümü anlatılmaktadır. Hesaba giriş dersinde, algoritma tasarlamak için bir uygulama dili kullanılmaktadır. Sebebi ise öğrencinin canını sıkan ve dikkatini en temel noktadan başka yönlere çeken detaylardan kurtarmaktır. Uygulama dili kullanılan Hesaba Giriş dersinin sonunda yapılan ölçümler, öğrencilerin sonuç veren ve doğru algoritmik tasarımlar yapabildiğini göstermiştir. Shackelford ve LeBlanc'ın kullandıkları uygulama dilinin başarıya ulaşması, yapılan çalışmanın etkili bir yöntem olacağını doğrular nitelikletedir[4].

Konuyla ilgili olarak Mansoor Al-A'Ali ve Mohammed Hamid, Design of an Arabic Programming Language (ARABLAN) adlı çalışmalarında, öğrencilerin Latince temelli bir programlama dilini öğrenebilmesi için en azından o dildeki komutların anlamlarını bilmesi gerektiğini ve konuşma dili İngilizce olmayan toplumlarda da bunun kolay bir iş dolmadığını vurgulamışlardır. Yaptıkları Arapça bir programlama dili çalışması ile uygulama dili ve öğrenenlerin konuşma dilinin aynı olmasının önemini belirtmişlerdir[5].

Nicolas Guibert ve Patrick Girard, Teaching and Learning with a Programming by Example System adlı çalışmalarında üniversite öğrencilerine zorunlu programlama kavramlarını öğretmek için soyut tanımlar yerine somut örnekler kullannan MELBA (Metaphor-Based Environment to Learn to Built Algoritms), adlı sistemi tasarlamışlardır. MELBA sisteminde kullanılan mecazi dil ile programlama mantığı öğreticisinde kullanılan uygulama dili birbirlerine benzemektedir[6].

Stephan Cooper ve arkadaşları, ALICE: A 3-D Tool for Introductory Programming Concepts adlı çalışmalarında, programlama öğretiminde animasyon kullanımını incelemişlerdir. Animasyon kullanımı örnekleri olan XTANGO, BALSA ve Pascal öğretiminde oldukça başarılı bir araç olan KAREL adlı örneklerden bahsetmişlerdir. ALICE adlı çalışmaları ile de yeni başlayanlar için bir programlama dili sunmuşlardır[7]. Animasyonla öğretim yaklaşımı, tasarlanan uygulama dilinden farklı olmakla birlikte öğretimin etkinliğini arttırmak amacıyla kullanılan farklı metotlardan bir tanesidir.

Victor Adamchik ve Ananda Gunawardena, A Learning Objects Approach to Teaching Programming adlı çalışmalarında klasik kitapların programlama dersleri için etkili olmadığını belirtmişler ve bunun yerine ders notu, kod örnekleri, çalışma soruları ve programlama kısımlarını içeren bir öğrenme nesnesi tanımlamışlardır[8]. Programlama mantığı öğreticisi, yardım düğmeleri ve programlama ipuçlarının aktif kullanılmasıyla üstte bahsedilen öğrenme nesnesiyle benzerlikler göstermektedir.

3. PROGRAMLAMA MANTIĞI ÖĞRETİCİSİ TASARIMI

Bu bölümde programlama mantığının teorik temelden pratiğe dönüştürülmesini sağlayan bir uygulama dilinin geliştirildiği öğretici isimli sistemin tasarımı ve çalışmasına yer verilmiştir.

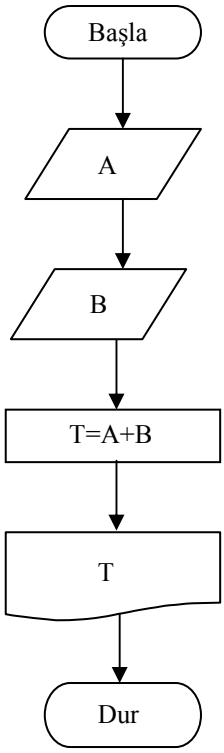
Öğretici sistemine kısaca PROGMAN denilmektedir. Programlama mantığı öğreticisi bundan böyle PROGMAN olarak anılacaktır.

Burada programlama mantığı öğreticisinin tasarımına geçmeden önce programlama mantığı ile ilgili bazı bilgilerin verilmesi faydalı olacaktır. Programlama mantığı eğitimi, programlama eğitiminde en önemli aşamalardan birisidir. Altta verilen örnek programlama mantığı öğretiminde kullanılan klasiklerden bir tanesidir. Teorik öğretim yöntemiyle PROGMAN kullanılan öğretim yönteminin karşılaştırılabilmesi açısından incelenmesi gerekmektedir.

Problem: Klavyeden girilecek iki sayının toplamını bulup yazdıran algoritmayı tasarlayın.

Çözüm: İki adet sayının sisteme girişi yapılmalıdır. Bu “okuma” işlemidir. Okunan iki sayının hafızada tutulabilmesi için iki alan (değişken), işlemin gerçekleştirilmesinden sonra ortaya çıkacak sonucun saklanması için de bir alan ayrılması gerekmektedir.

Şekil 3.1’de akış diyagramları kullanılarak çözüm gerçekleştirilmiştir.



Şekil 3.1. Akış diyagramları

Alttaki şekilde de doğruluk tablosu kullanılarak gerçekleştirilen çözüm gösterilmektedir.

A	B	T	Ekran
2	3	5	5

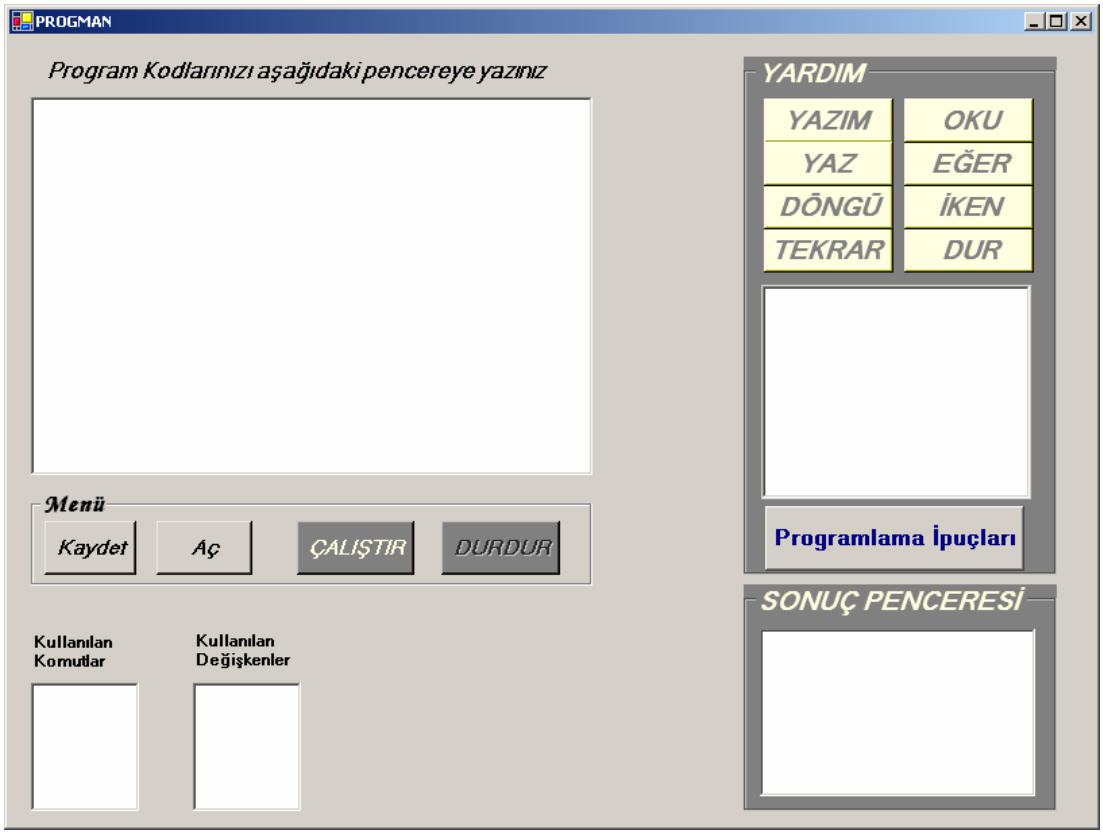
Şekil 3.2. Doğruluk tablosu

Altta ise akış diyagramları ile çözülmüş problemin uygulama komutları kullanılarak ve bir sıra takip edilerek aktarılmış hali aşağıdadır.

BAŞLA
OKU A
OKU B
T=A+B
YAZ T
DUR

Algoritma öğretimi sırasında kullanılan üstteki örnek, sadece teorik olarak tahtada veya kağıt üzerinde gösterilmektedir. Eker (2004), Algoritmayı Anlamak adlı çalışmasında da belirtilen çözümü deneme ve çözümü geliştirme işlem basamakları uygulanamamaktadır. Dolayısıyla programlama mantığı öğretiminde eksiklikler ve gecikmeler ortaya çıkmaktadır. Gerçekleştirilen çözüm her ne kadar kağıt üzerinde doğruluk tablosu kullanılarak çalıştırılsa bile, uygulama ile eş değer tutulamayacağı da bir gerçektir. Yapılan bir çözümün uygulanması ise, programlama mantığı öğretimi aşamasından sonraki adımda herhangi bir derleyici (Pascal, C, C++ vb.) ile yapılabilmektedir.

Yukarıdaki örneklerde de belirtildiği gibi programlama mantığı öğretimi önemli bir aşamadır. Programlamaya yeni başlayanlar için, öğrenmeyi kolaylaştırmayı hedefleyen PROGMAN, Windows ortamında çalışmaktadır. Dosya boyutunun oldukça küçük olması sebebiyle mail yoluyla yada bir disketle rahatlıkla taşınabilir. PROGMAN, görselliği ön planda tutmakta ve Türkçe olduğu için de olumsuz ön yargılardan sıyrılmaktadır. Tahtada veya kağıt üzerinde yapılan işleri bilgisayar ortamına taşımaktadır. Bu yönü itibariyle de, bilgisayar destekli eğitim materyalidir. Öğrenciyi kağıt ve kalemden kurtardığı için, çalışmayı zevkli hale getirecek, Türkçe olduğu için de öğrenciyi korkutmayacaktır. Bunların sonucu olarak da öğrencinin tamamen soyut kavramlarla çalışmasını önlendiği ve öğrenci üzerindeki zihinsel yükü azalttığı için başarıyı olumlu yönde etkileyecektir. Şekil 3.3'de PROGMAN'ın bir görüntüsü verilmiştir.



Şekil 3.3. Progman'ın genel bir görüntüsü

Yukarıda da görüldüğü gibi ara yüz, kavramların ve sonuçların anlaşılmasını kolaylaştırmak amacıyla tasarlanmıştır. Bu anlamda, ara yüz, 5 adet pencere 12 düğmeden oluşmuştur.

Sol üst köşede editör penceresi, sağda yardım penceresi ve her bir komuta ait yardım metinlerini gösteren düğmeler, altta ise yazılan programları çözümleyen pencereler bulunmaktadır. Çözümleme pencereleri, kullanılan komutlar, kullanılan değişkenler, Eğer ifadesi pencereleridir. Yukarıda bahsedilen pencereler kullanılan komutlara bağlı olarak karşımıza çıkmaktadır. Örneğin Eğer ifadesi kullanıldıysa Eğer çözümleme penceresi, görüntülenecektir.

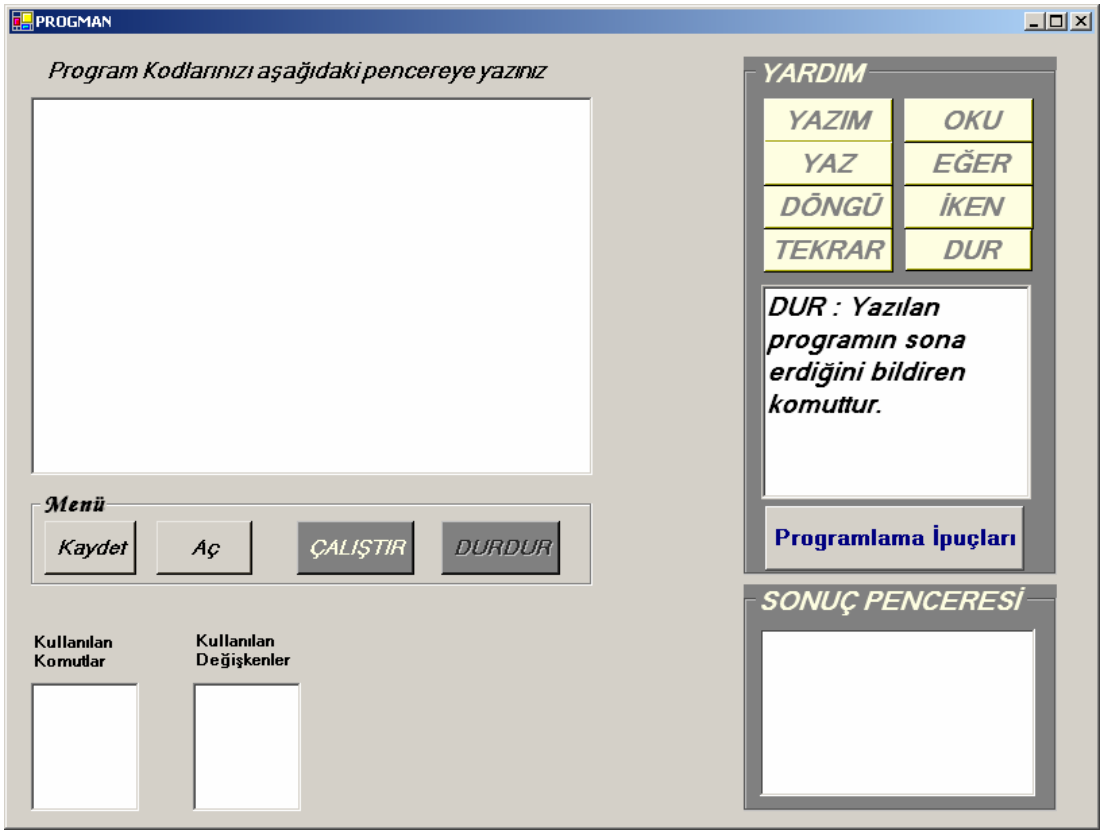
Sağ tarafta bulunan yardım düğmelerinin her biri bir komutu temsil etmektedir hangi komut veya yazım kuralı ile ilgili bilgi alınmak isteniyorsa o düğme tıklanır. Alttaki pencerede de ilgili yardım konuları görüntülenir.

Sol üst köşedeki editör penceresinde, program çalıştığında imleç varsayılan olarak konumlanır ve programlar buraya yazılır. Editör penceresinin hemen altında bulunan Kaydet düğmesine tıklandığında yazılan program. RTF formatında kaydedilir. Aç düğmesi ile de daha önceden yazılan ve kaydedilen programlar editör penceresine getirilir. Hemen yanındaki Çalıştır düğmesi yazılan programı derler ve sonuçlarını görmemizi sağlar. Durdur düğmesi ise programın çalışmasını sona erdirir. Durdur düğmesine basılmadan önce yazılan program mutlaka kaydedilmelidir.

PROGMAN'ın açılışında Şekil 3.4'deki mesaj görüntülenir.

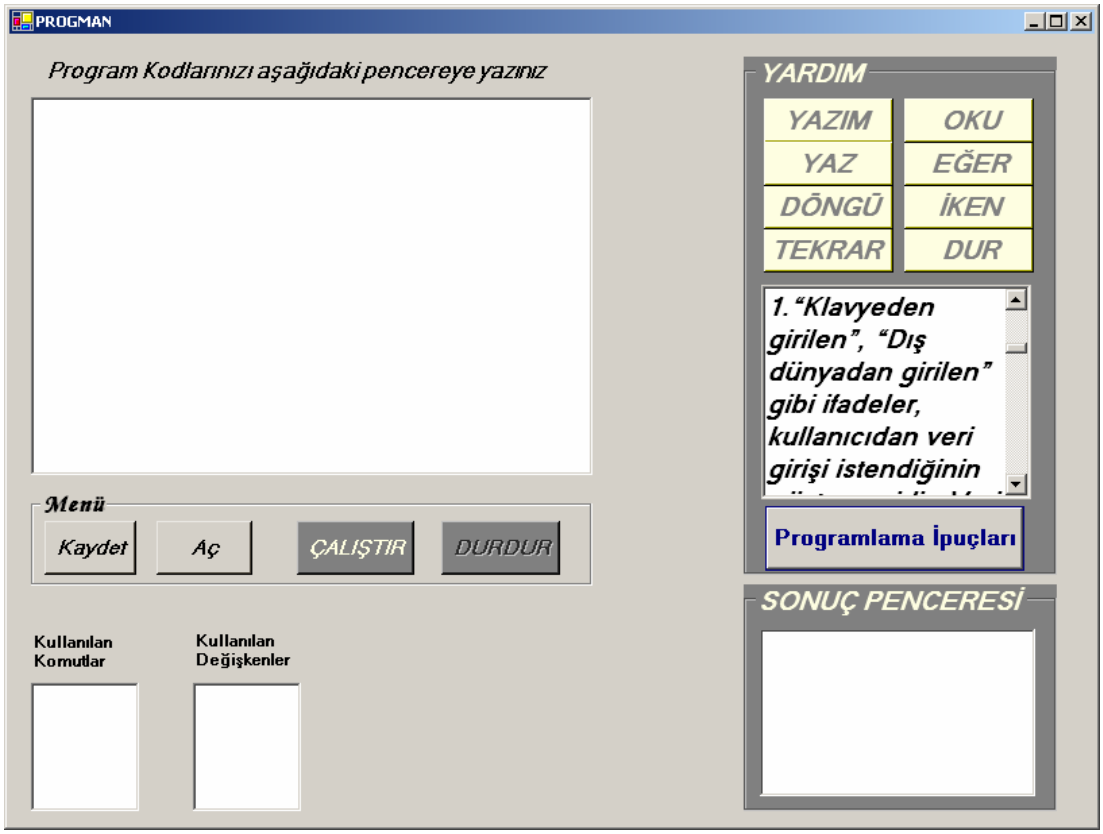
Şekil 3.4. Progman açılış mesajı görüntüsü

Yukarıda görüntülenen mesajın anlamı, PROGMAN'ın özeliklerinin tam olarak kullanılabilmesini amaçlamaktadır. Yardım düğmeleri, genel yazım kuralları ve her bir komutun kullanımı ile ilgili bilgiler vermektedir. Yardım konularının Türkçe olmasıyla, etkisinin de artacağı düşünülmektedir. Aktif olarak yardım kullanımı programlamada sıkça karşılaşılan hatalara, anında çözüm bulabilmek açısından oldukça önem taşımaktadır. Şekil 3.5'te ise yardım düğmelerinden bir tanesinin görüntüsü bulunmaktadır.



Şekil 3.5. Progman'ın yardım kullanımı görüntüsü

Ayrıca Programlama İpuçları düğmesiyle, verilen bir problemin anlaşılmasını dolayısıyla çözümünü kolaylaştıracak ipuçlarını sunmaktadır. Bu ipuçları, problem çözümleme yöntemleridir. Problem çözümleme ise, problemde geçen her bir kelimenin komut olarak karşılığının bulunabilmesidir. Eğer bir problem anlaşılmış ve çözümlenmiş ise geriye kalan tek şey kodlamanın yazılmasıdır. Programlama ipuçları düğmesinin kullanımı ise Şekil 3.6'da gösterilmiştir.



Şekil 3.6. Progman'ın programlama ipuçları görüntüsü

PROGMAN'da veri girişi ise Şekil 3.7'de gösterildiği gibi yapılmaktadır.

Şekil 3.7. Progman'da veri giriş penceresi

PROGMAN, zihinsel olarak yapılan işlemleri görselleştiren bir sistemdir. Problem çözümüne ve yazılacak programlara yardımcı olur. Kullanıcılara kendi dillerinde, emir kipinden oluşan uygulama komutlarıyla program yazabilmeyi sağlar.

Programda değişken tanımlama zorunluluğu yoktur. Kullanılan komutlar, programların temelini oluşturan kavramlardır. Veri yapıları, diziler, özyineleme gibi karmaşık ve öğrenilmesi zor konuları içermediği için, kullanıcının dikkatini, en temel kavramlar üzerinde yoğunlaştırır. Yazım kuralları veya kullanışları farklı olmakla birlikte aynı görevi yapan bu kavramların tüm programlama dillerinde ortak olarak kullanılması ise belirleyici bir özelliktir.

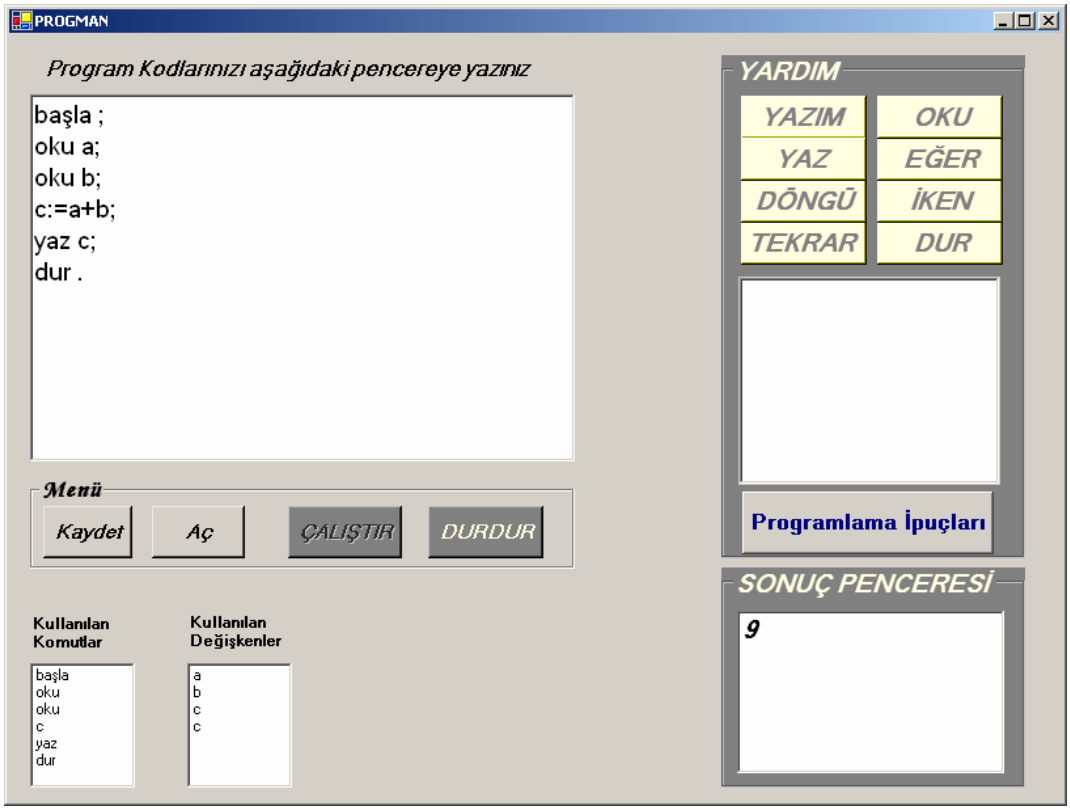
Kullanıcılar bu genel komutların yapısını ve kullanılışını Türkçe komutlarla daha kolay öğrenecekler, bir programlama dili öğrenimine başladıklarında ise PROGMAN'da öğrendiklerini, kullandıkları dile aktararak daha da başarılı olacaklardır. Bu bağlamda kullanılan komutlar Oku, Yaz, Eğer, Döngü, İken Tekrarla' dır.

3.1. Örneklerle Tanıtım

Problem 1: İki sayının toplamını bulduran bir algoritma tasarımı ile başlanırsa, aşağıdaki kodlar kullanılacaktır.

```
başla ;
oku a;
oku b;
c:=a+b;
yaz c;
dur .
```

Yukarıdaki programa girdi olarak 4 ve 5 değerleri verildiğinde ise sonuç Şekil 3.8' deki gibi görüntülenecektir.

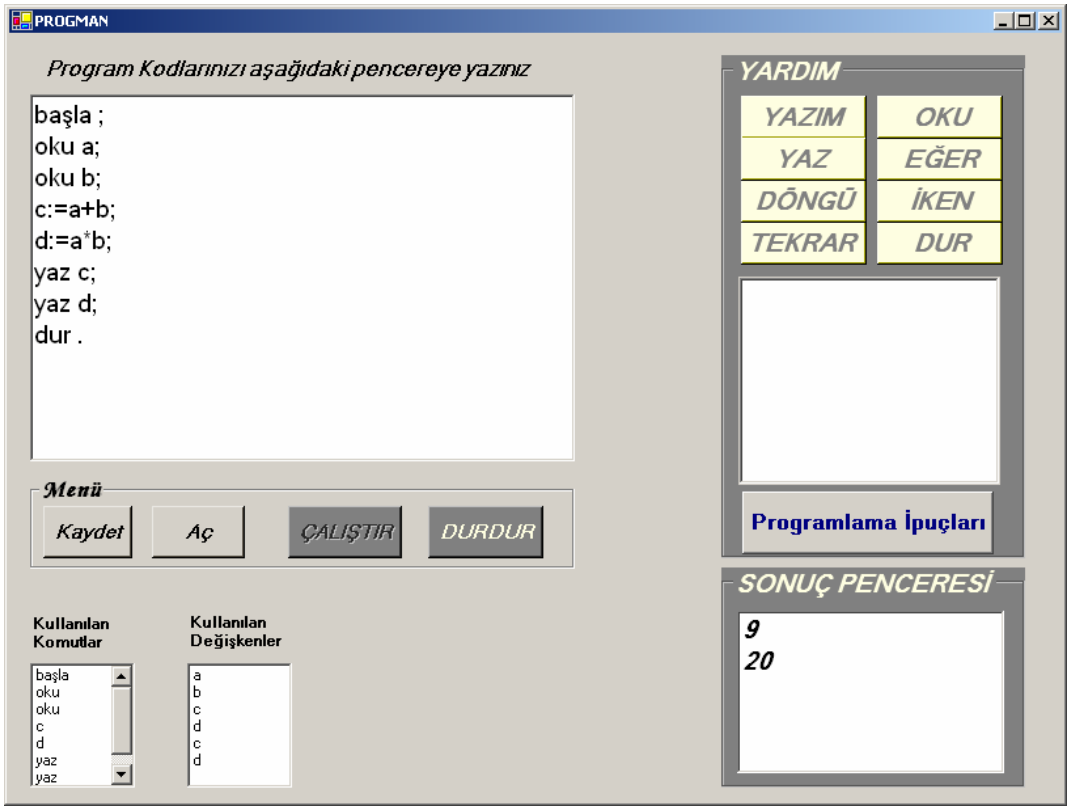


Şekil 3.8. Progman'ın çalışma görüntüsü

Problem 2: Kullanıcıdan istenen iki adet tamsayının toplamını ve çarpımını hesaplatan bir algoritma tasarımı istendiğinde aşağıdaki komutlar kullanılacaktır.

başla ;
oku a;
oku b;
c:=a+b;
d:=a*b;
yaz c;
yaz d;
dur .

Programa girdi olarak sırasıyla 4 ve 5 değerleri girildiğinde sonuç Şekil 3.9'daki gibi görüntülenecektir.

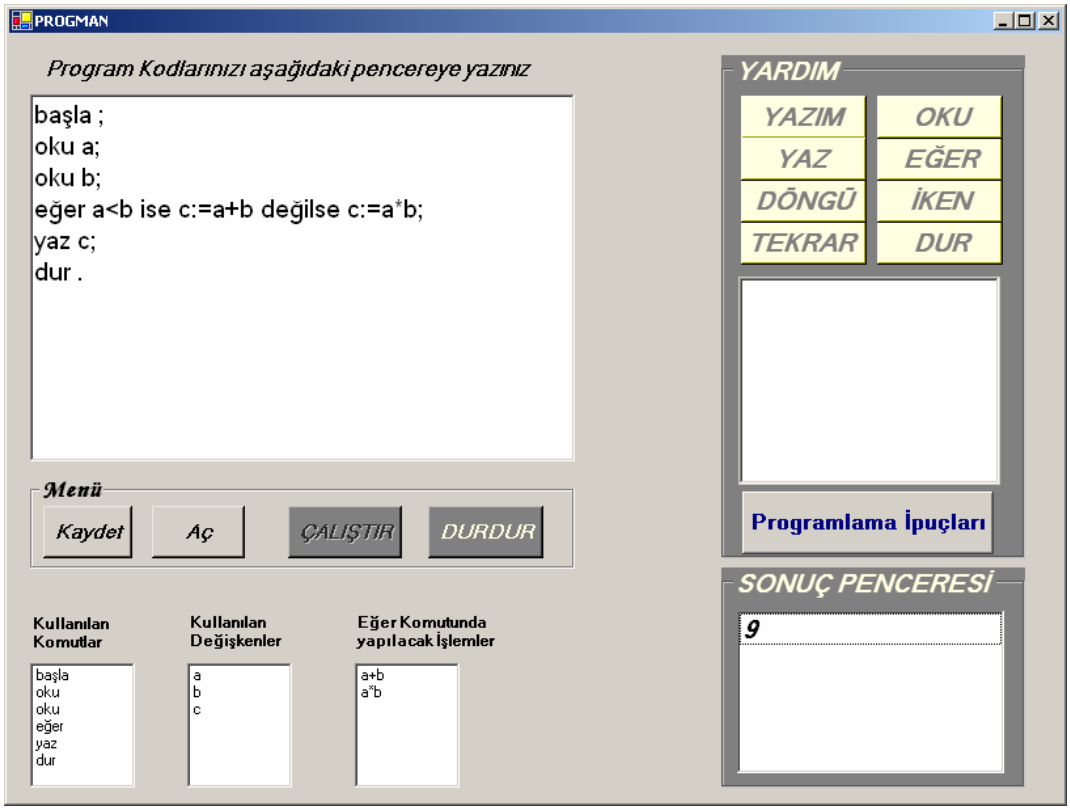


Şekil 3.9. Progman'ın çalışma görüntüsü

Problem 3: Girilen iki sayıdan, birinci sayı, ikinci sayıdan küçük ise sayıların toplamını hesaplayan, büyük ise de sayıların çarpımını hesaplatan bir algoritma tasarımı istenirse eğer aşağıdaki kod satırları çözümü oluşturacaktır.

başla ;
oku a;
oku b;
eğer a<b ise c:=a+b değilse c:=a*b;
yaz c;
dur .

Programa girdi olarak sırasıyla 4 ve 5 değerleri girildiğinde sonuç Şekil 3.10' daki gibi görüntülenecektir.

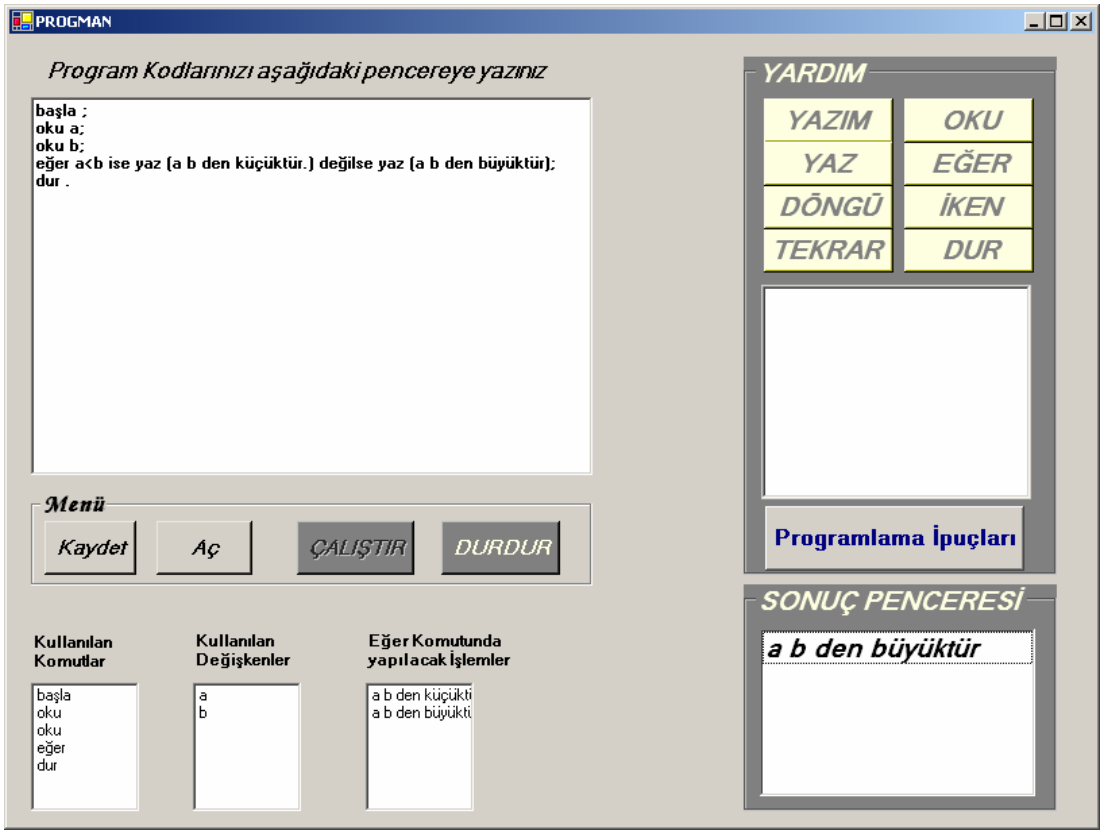


Şekil 3.10. Progman'ın çalışma görüntüsü

Problem 4: Girilen iki sayıyı karşılaştırarak küçük veya büyük olduğunu, ekrana yazdıran bir algoritma tasarımı istenirse eğer aşağıdaki kod satırları çözümü oluşturacaktır. Program aşağıdaki pencere görüntüsünü oluşturacaktır.

başla ;
oku a;
oku b;
eğer a<b ise yaz (a b den küçüktür.) değilse yaz (a b den büyüktür);
dur .

Programa girdi olarak sırasıyla 5 ve 4 değerleri girildiğinde sonuç Şekil 3.11' deki görüntülenecektir.

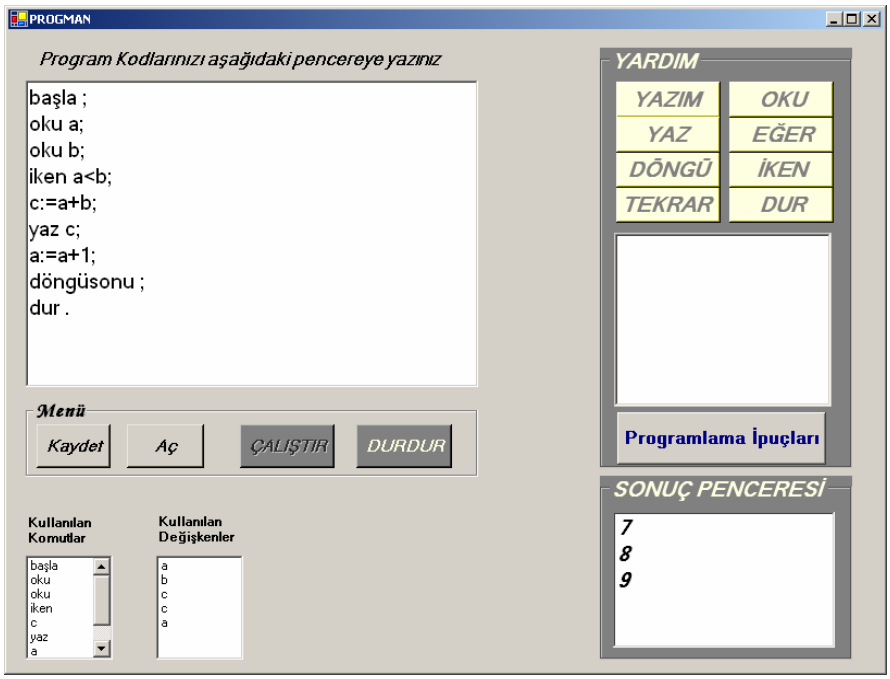


Şekil 3.11. Progman'ın çalışma görüntüsü

Problem 5: Klavyeden iki adet sayı girilecektir. Küçük olan sayı birer artırılacaktır. Küçük sayı büyük sayıyı yakalayıncaya kadar, her bir adımda sayıların toplamını bulup, ekrana yazdıran bir algoritma tasarımı istenirse eğer aşağıdaki kod satırları çözümü oluşturacaktır.

başla ;
oku a;
oku b;
iken a<b;
c:=a+b;
yaz c;
a:=a+1;
döngüsonu ;
dur .

Programa girdi olarak sırasıyla 2 ve 5 değerleri girildiğinde sonuç Şekil 3.12' deki gibi görüntülenecektir.

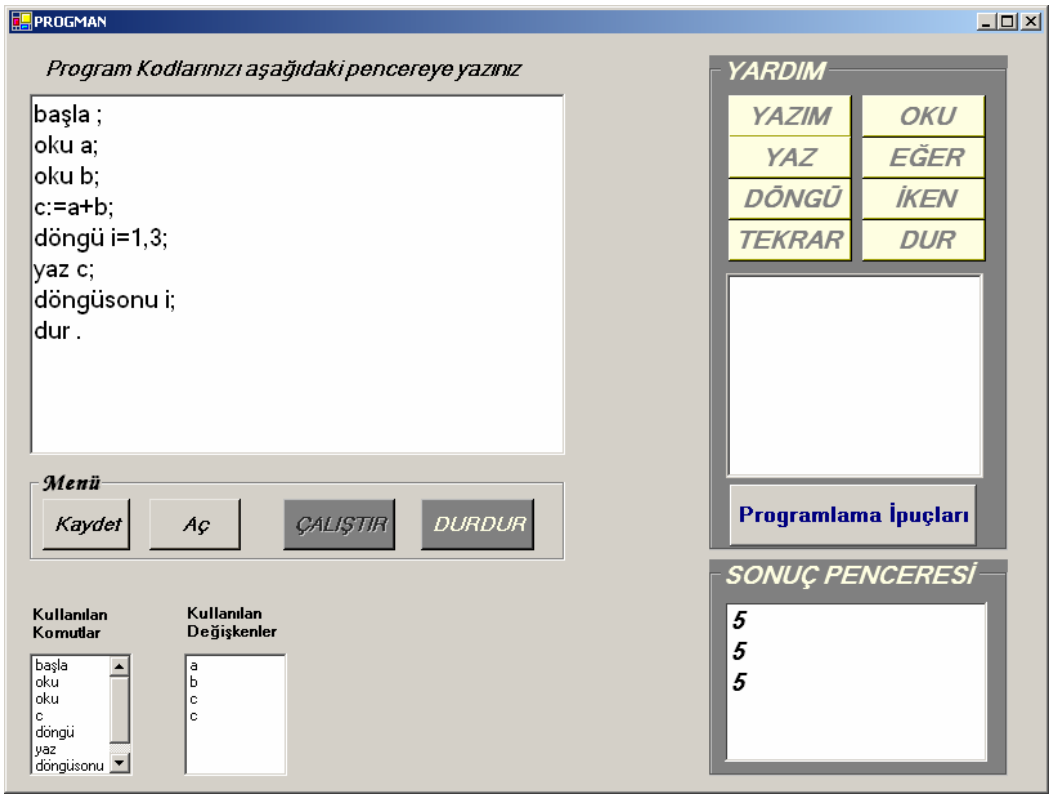


Şekil 3.12. Progman'ın çalışma görüntüsü

Problem 6: Girilen iki adet sayının toplamını, ekrana 3 kez yazdıran bir algoritma tasarımı istenirse eğer aşağıdaki kod satırları çözümü oluşturacaktır.

başla ;
oku a;
oku b;
c:=a+b;
döngü i=1,3;
yaz c;
döngüsonu i;
dur .

Programa girdi olarak sırasıyla 2 ve 3 değerleri girildiğinde sonuç Şekil 3.13' deki gibi görüntülenecektir.



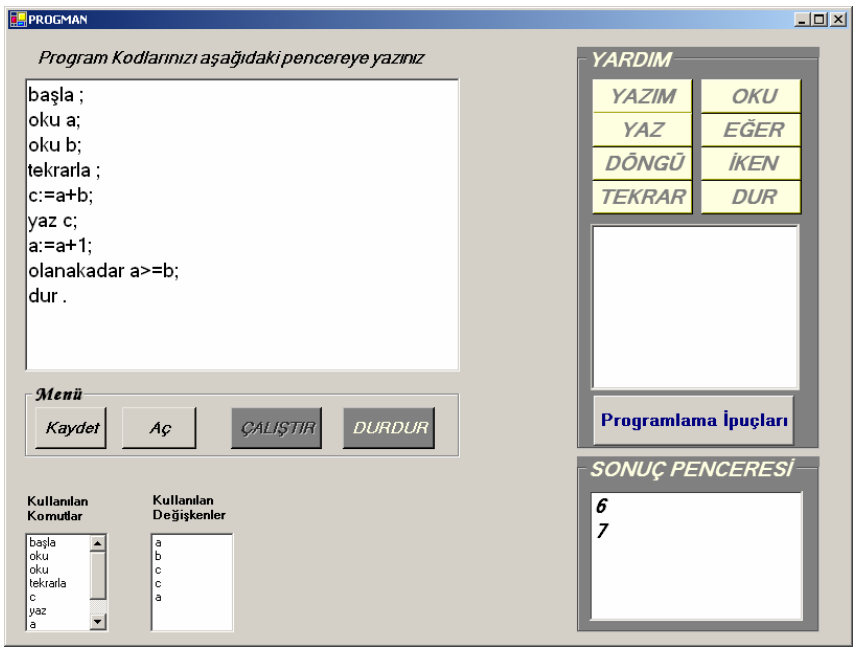
Şekil 3.13. Progman'ın çalışma görüntüsü

Problem 7: Klavyeden iki adet sayı girilecektir. Küçük olan sayı birer artırılacaktır. Küçük sayı büyük sayıdan büyük veya eşit olana kadar, her bir adımda sayıların toplamını bulup, ekrana yazdıran bir algoritma tasarımı istenirse eğer aşağıdaki kod satırları çözümü oluşturacaktır.

başla ;
oku a;
oku b;
tekrarla ;
c:=a+b;
yaz c;

a:=a+1;
olanakadar a>=b;
dur .

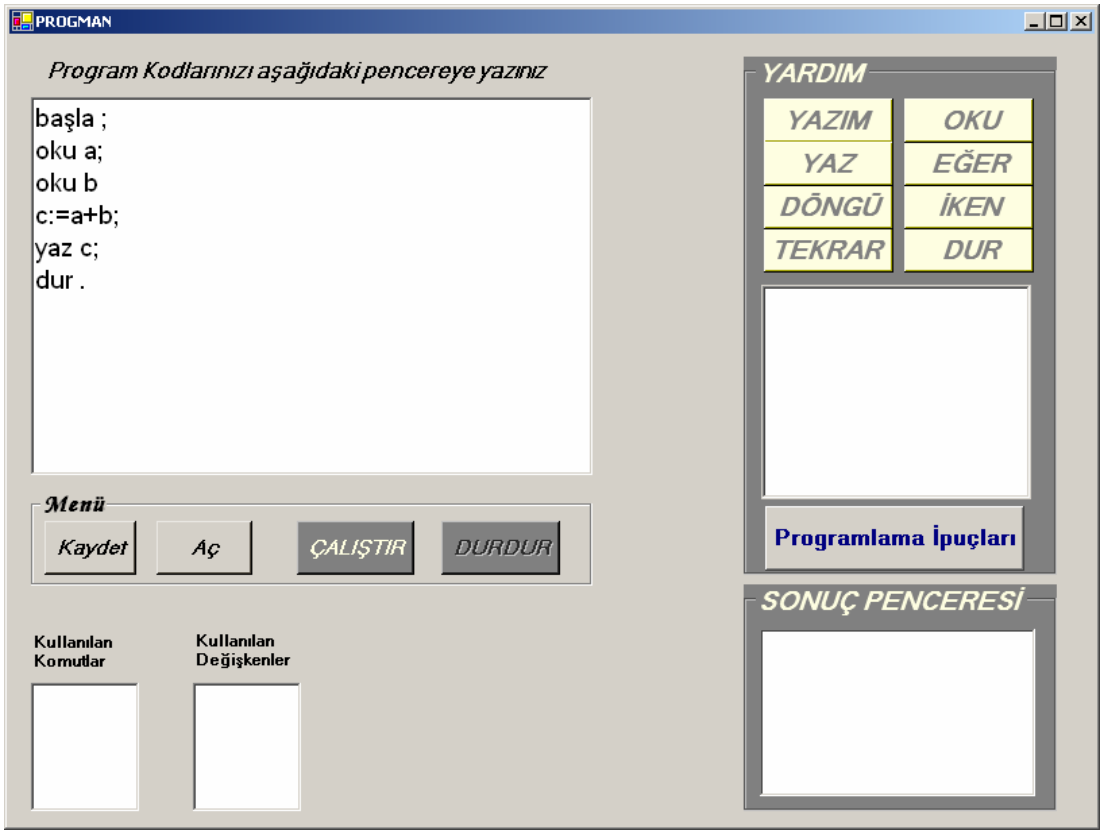
Programa girdi olarak sırasıyla 2 ve 4 değerleri girildiğinde sonuç Şekil 3.14' deki gibi görüntülenecektir.



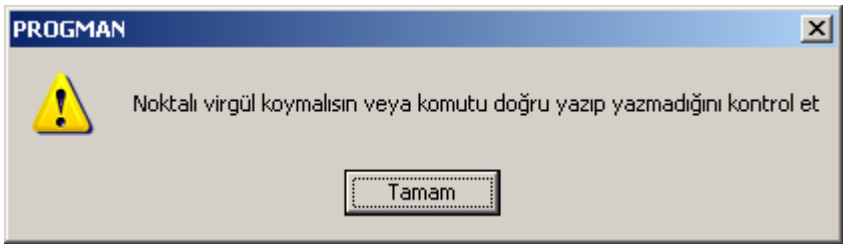
Şekil 3.14. Progman'ın çalışma görüntüsü

3.2. Hata Kontrolü

Şekil 3.15' de görülen, iki sayının toplamını bulduran algoritma tasarımında, 3. satırın sonuna ";" konmaz ise Şekil 3.16' da görülen uyarı mesajı ekrana gelecektir.

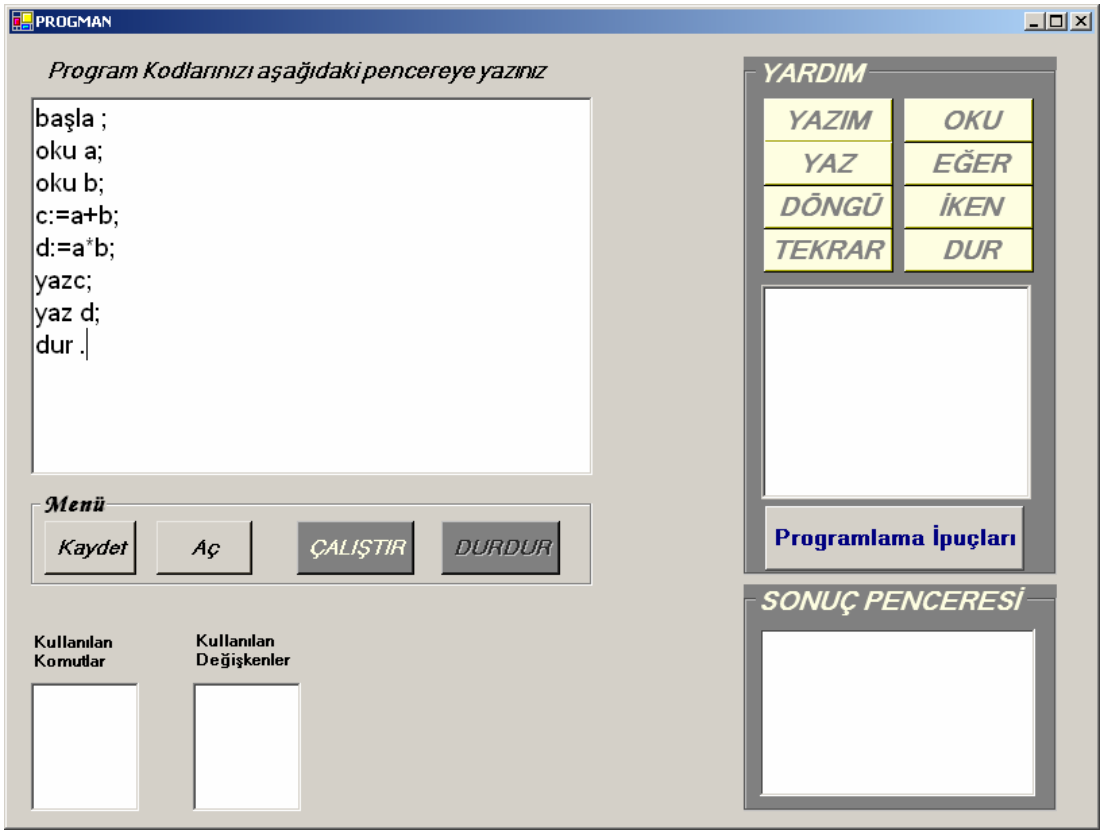


Şekil 3.15. Algoritma tasarım görüntüsü

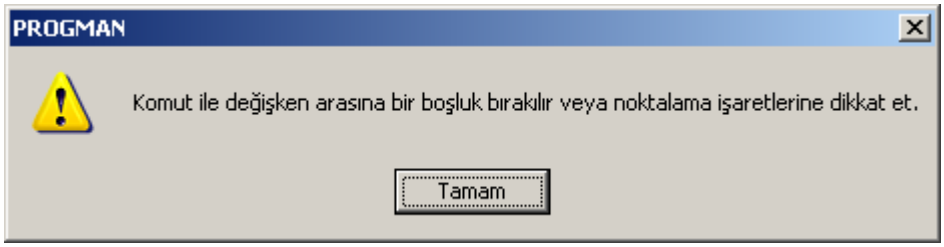


Şekil 3.16. Hata mesajı

Şekil 3.17' de görülen iki adet tamsayının toplamını ve çarpımını hesaplatan bir algoritma tasarımında, 6. satırda komut ile değişken arasında boşluk bırakılmazsa Şekil 3.18' da görülen uyarı mesajı ekrana gelecektir.

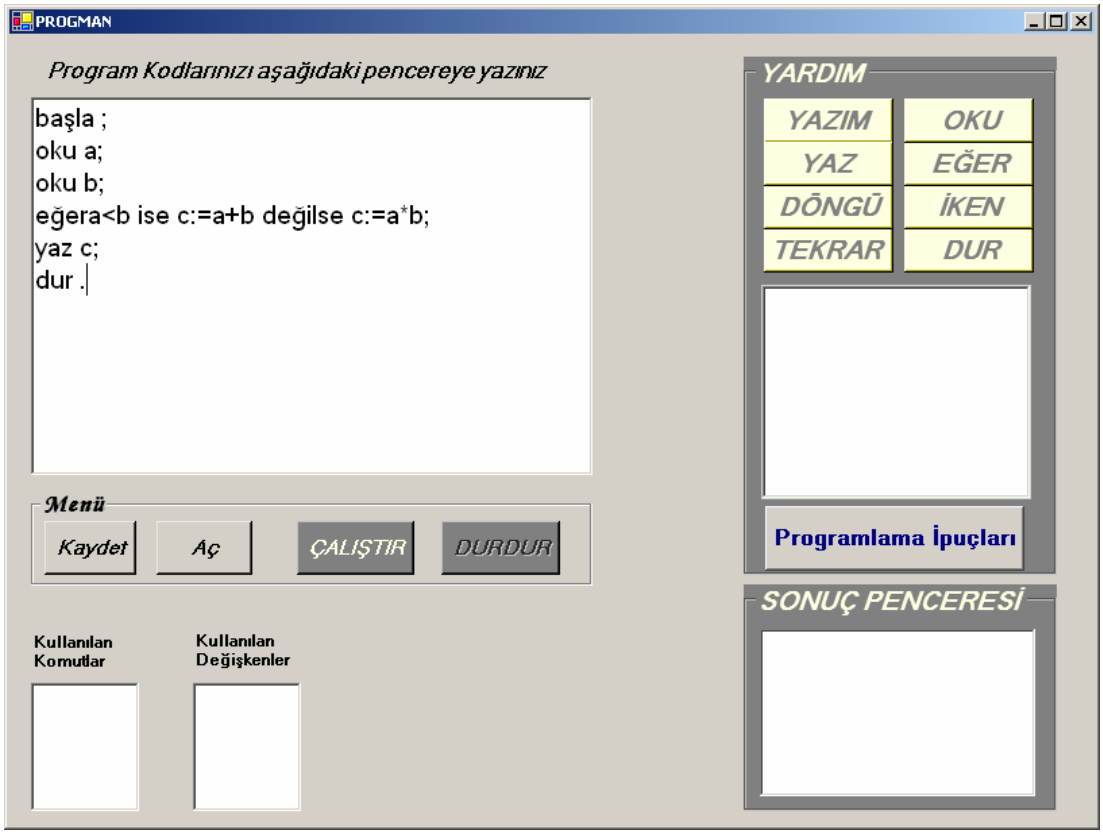


Şekil 3.17. Algoritma tasarım görüntüsü

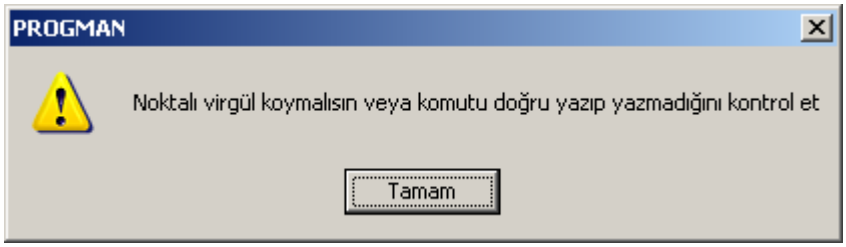


Şekil 3.18. Hata mesajı

Şekil 3.19' da görülen, girilen iki sayıdan, birinci sayı, ikinci sayıdan küçük ise sayıların toplamını hesaplayan, büyük ise de sayıların çarpımını hesaplatan bir algoritma tasarımında eğer komutunun yazımında yapılan yanlış sonucu Şekil 3.20' de görülen uyarı penceresi ekrana gelecektir.

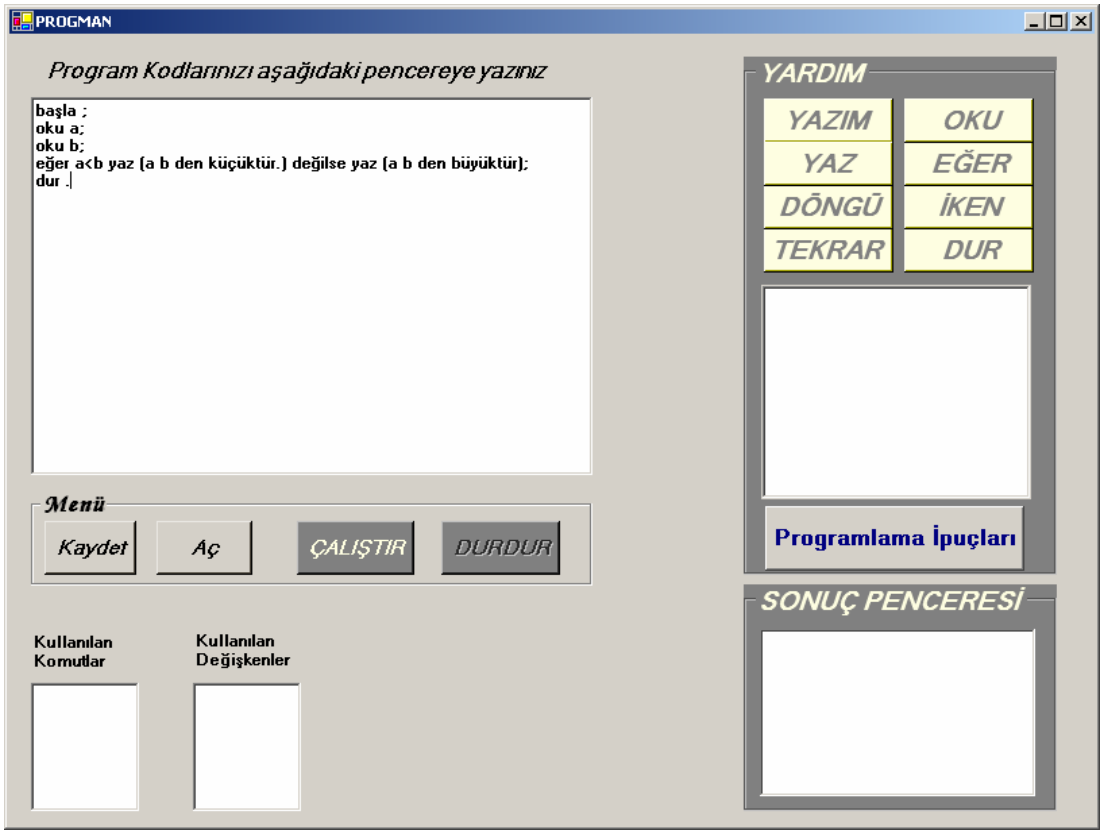


Şekil 3.19. Algoritma tasarım görüntüsü

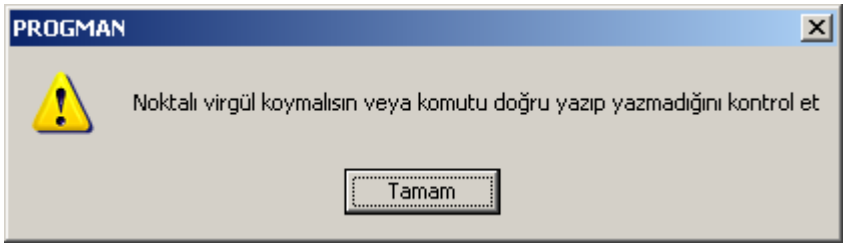


Şekil 3.20. Hata mesajı

Resim 3.21' de görülen, girilen iki sayıyı karşılaştırarak küçük veya büyük olduğunu, ekrana yazdıran bir algoritma tasarımında, eğer komutunun değilse ifadesinden sonra bir komut yada deyim yazılmazsa, Resim 3.22' de görülen uyarı penceresi ekrana gelecektir.

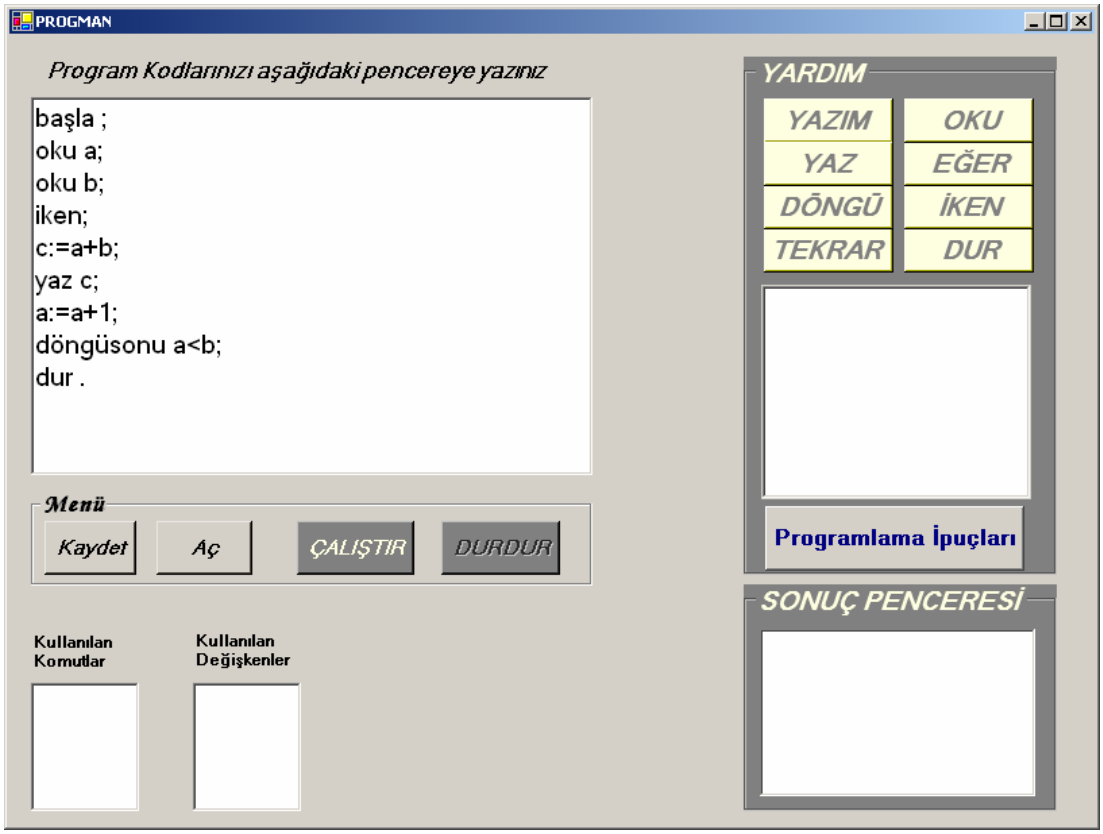


Şekil 3.21. Algoritma tasarım görüntüsü

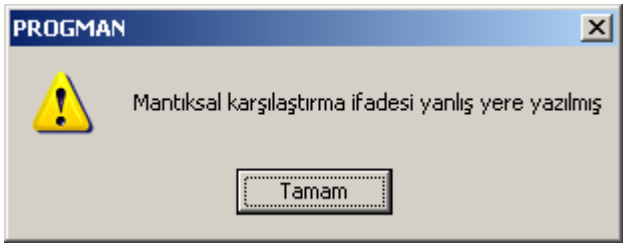


Şekil 3.22. Hata mesajı

Şekil 3.23’ de görüldüğü gibi, klavyeden iki adet sayı girilecektir. Küçük olan sayı birer artırılacaktır. Küçük sayı büyük sayıyı yakalayıncaya kadar, her bir adımda sayıların toplamını bulup, ekrana yazdıran algoritma tasarımında karşılaştırma ifadesi iken komutuyla birlikte yazılması gerekirken yanlış yere yazılması sonucunda Şekil 3.24’ de görülen uyarı penceresi ekrana gelecektir.

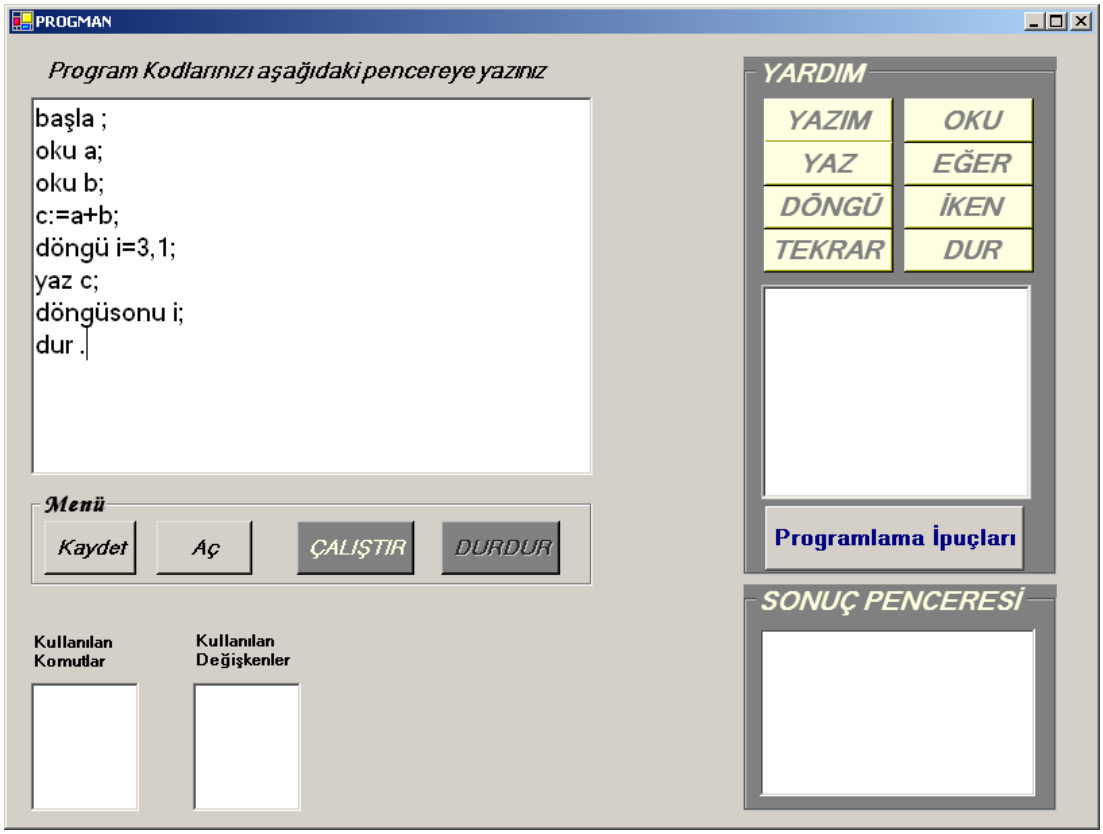


Şekil 3.23. Algoritma tasarım görüntüsü



Şekil 3.24. Hata mesajı

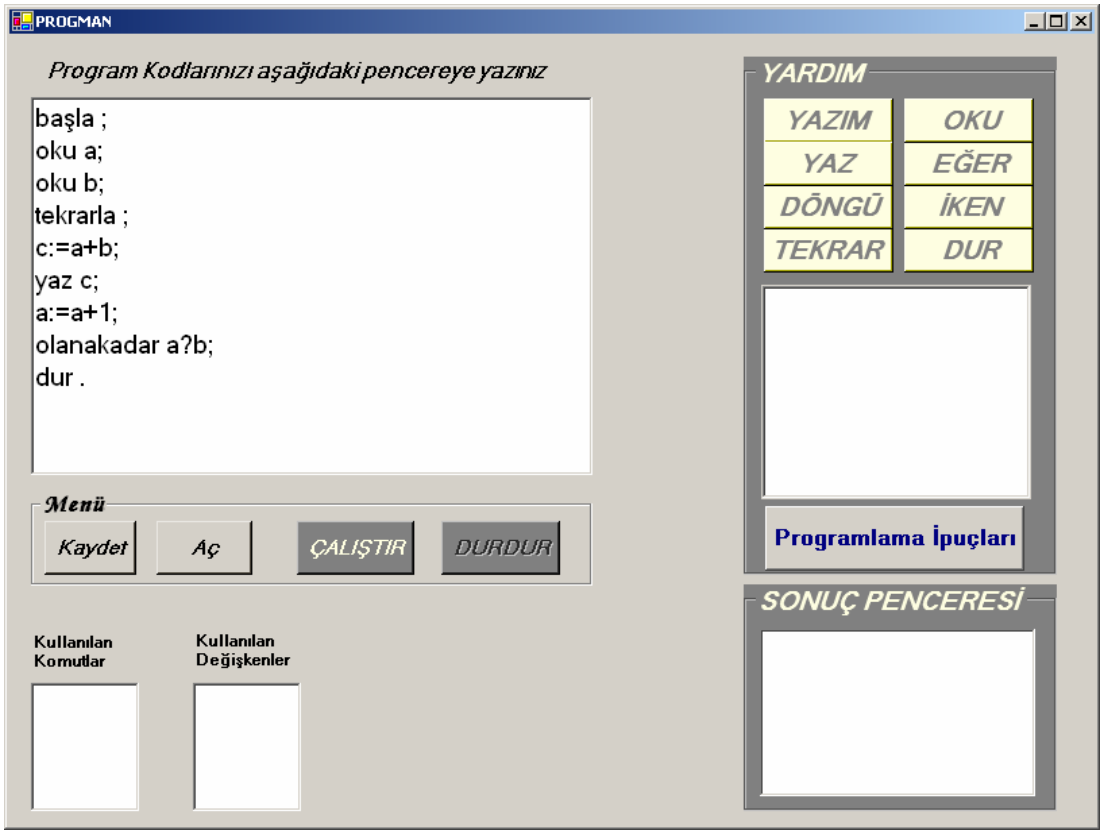
Şekil 3.25' de görülen, girilen iki adet sayının toplamını, ekrana 3 kez yazdıran bir algoritma tasarımı istendiğinde, döngü komutunun başlangıç ve bitiş değerlerinin yanlış verilmesi sonucunda, Şekil 3.26'da görülen uyarı penceresi ekrana gelecektir.



Şekil 3.25. Algoritma tasarım görüntüsü

Şekil 3.26. Hata mesajı

Şekil 3.27' de görüldüğü gibi, klavyeden iki adet sayı girilecektir. Küçük olan sayı birer artırılacaktır. Küçük sayı büyük sayıdan büyük veya eşit olana kadar, her bir adımda sayıların toplamını bulup, ekrana yazdıran bir algoritma tasarımı esnasında mantıksal karşılaştırma ifadesinin yanlış yazılması sonucunda Şekil 3.28' de görülen uyarı penceresi ekrana gelecektir.



Şekil 3.27. Algoritma tasarım görüntüsü

Şekil 3.28. Hata mesajı

4. SONUÇ VE ÖNERİLER

Bu çalışmada, programlama mantığı öğretimi için kullanılabilir bir eğitim materyali geliştirilmiştir. Geliştirilen bu materyal, programlama mantığı öğretimi sürecinde bir takım zorluklar yaşanan bir problemin çözüm basamaklarından olan çözümü deneme ve çözümü geliştirme basamaklarını kolaylaştırmayı amaçlamaktadır.

Söz konusu çözüm basmakları, teorik yöntemle yapılan algoritma tasarımlarında, kullanılamamaktadır. Diğer bir deyişle öğrenci yaptığı yanlışı görememekte, göremediği için de yaptığı çözümün doğruluğundan hiçbir zaman emin olamamaktadır. PROGMAN aracılığıyla söz konusu çözüm basamakları görsel ve kullanılabilir hale getirilmiştir.

Örneğin, girilen iki adet sayının toplamını, ekrana 3 kez yazdıran bir algoritma tasarımı istenirse eğer aşağıdaki kod satırları çözümü oluşturacaktır.

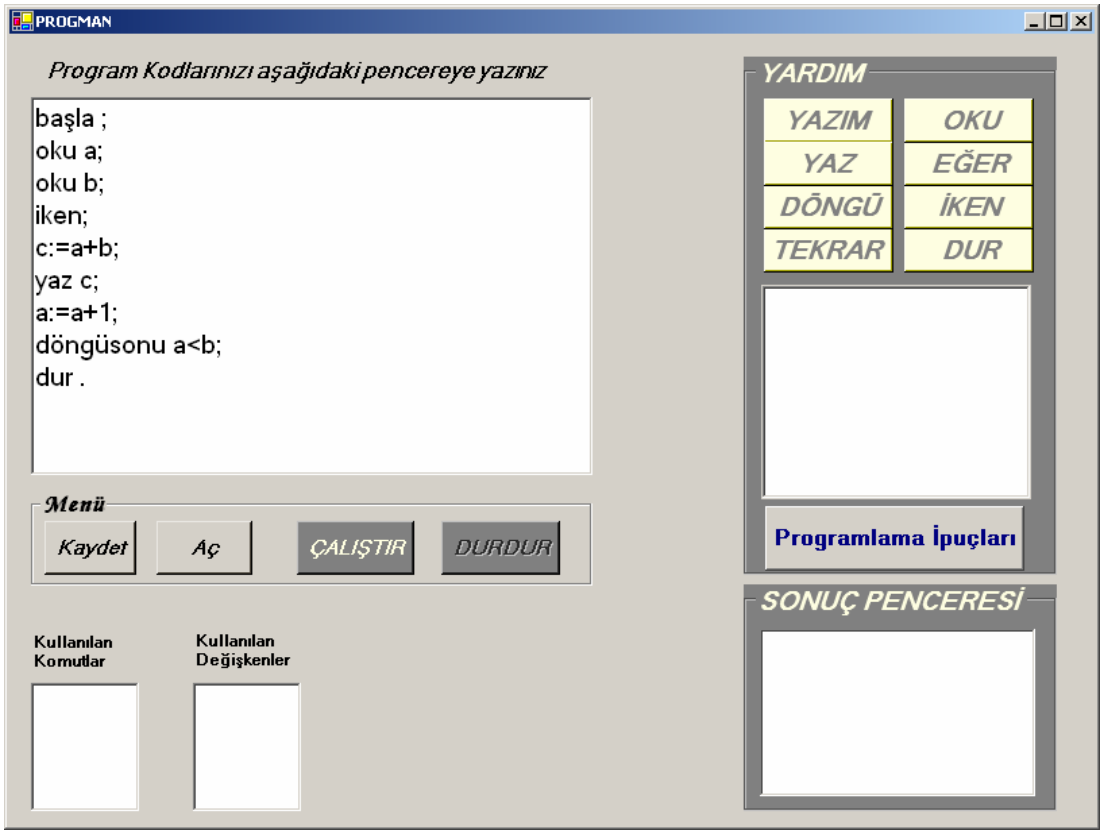
```
başla ;
oku a;
oku b;
c:=a+b;
döngü i=1,3;
yaz c;
döngüsonu i;
dur .
```

Üstteki uygulama kodlarının nasıl çalıştığını anlatabilmek için bölüm 3'te de belirtildiği gibi doğruluk tablosu kullanılmaktadır.

Doğruluk tablosu kullanılarak gerçekleştirilen çözümün bazı sakıncaları bulunmaktadır. Bu çözüm yöntemi tahtada veya kağıt üzerinde yapılmakta ve soyut düşünebilme yeteneği üzerine kurulmuştur. Öğrencinin yanlış yapması durumunda ders öğretmeni veya öğretim elemanı dışında herhangi bir uyarıcı veya hata mesajı

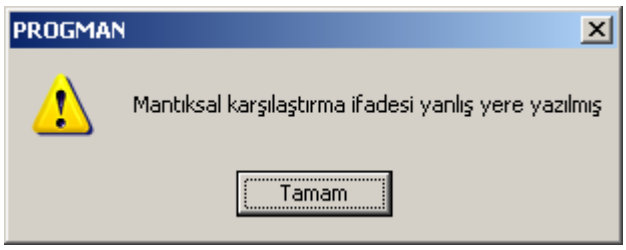
devreye girmemektedir. Dolayısıyla öğrencinin tek başına çalışmasını destekleyen nitelikte bir yöntem değildir. Programlamaya yeni başlayanlar için, sürekli bir başkasının desteğine ihtiyaç duymasına neden olmaktadır. Bununla birlikte, doğru sonuca yönlendirmediği için moral bozukluğuna neden olabilmektedir.

PROGMAN' ın kullanımıyla üstte bahsedilen dezavantajlar ortadan kalkacak teoriyi pratiğe dönüştürdüğü için, çalışmayı zevkli hale getirecektir. Bu yönüyle PROGMAN, bireysel farklılıkların da gözetildiği bir çalışma ortamı sunmaktadır.



Şekil 4.1. Progman'ın çalışma görüntüsü

Şekil 4.1' de de görüldüğü gibi kağıt üzerinde veya tahtada yapılabilen örnekler bilgisayar ortamına aktarılmıştır. Uygulamalı bir çalışma ortamının yanı sıra Şekil 4.2' de de görüldüğü gibi yapılan yanlışlar anında kullanıcıya bildirilmektedir.



Şekil 4.2. Progman'da hata mesajı

Yapılan araştırmalar, programlama mantığı öğretiminde, bir uygulama dili kullanımının başarıyı arttırdığını desteklemektedir. Yapılan bu çalışmada ise dili Türkçe olan, bir uygulama dili tasarlanmıştır. Dil olarak Türkçe kullanılması, yabancı dile olan önyargıları ve yabancı dil eksikliğini ortadan kaldıracak, öğretim materyali olarak bir uygulama dilinin kullanılması da çok sayıda komut ve özellik içermesinden dolayı öğrencilere oldukça karmaşık gelen ve öğretim sırasında aksaklıklara yol açan programlama dillerinin kullanımını da azaltmaktadır.

Oldukça basit bir ara yüze sahip olan PROGMAN, öğretim basamağının en başında öğrenciyi olumlu yönde güdüleyecektir. Öğrenci, PROGMAN'ı kullanarak, yaptığı çalışmalardan olumlu sonuçlar elde edecek, bu başarılar da ilerisi için öğrenci motivasyonunu attıracaktır. Başarının verdiği hazzı tadan bir öğrenci için de sonuçlar hep iyiye doğru gidecektir. Bunun yanı sıra Türkçe olarak tasarlanan uygulama dilinin, diğer dillere de kısa bir sürede uyum sağlayabilir olması, yapılan çalışmaya, farklı ülkelerde de uygulanabilirlik özelliği katmaktadır.

Yapılan çalışmanın daha ileriye götürülebilmesi için, kullanılan her bir komut için çözümleme pencereleri yapılmalıdır. Bununla birlikte kullanılan değişkenlerin de durum değişikliklerinin izlenebildiği bir veya birden fazla pencerenin de tasarlanması programın etkinliğini arttıracaktır. Ayrıca, yazılan programların satır satır çalıştırılabilmesine olanak sağlayan düzenlemelerin yapılması da problemlerin anlaşılmasını kolaylaştıracaktır. Ayrıca fiziksel sınırlamaların ortadan kaldırılması açısından, programın web üzerinde çalışan bir versiyonunun tasarlanması da uygun olarak değerlendirilmektedir.

Yapılan çalışmayı bir her yönüyle bir öğretim materyali haline getirebilmek için konu anlatımları da eklenebilir. Bu anlatımların da animasyon teknikleri ile donatılması ayrıca katkı sağlayacaktır.

KAYNAKLAR

1. Eker, M., "Algoritmayı Anlamak", ***Nirvana***, Ankara, 24-25(2004).

2. Kurnaz, S., "Veri Yapıları ve Algoritma Temelleri",***Papatya Yayıncılık***, İstanbul, 16-20 (2000).

3. Ramadhan, H.A., "Programming by Discovery", ***Journal of Computer Assisted Learning***, 16:83-93(2000).

4. Shackelford,R.L. and LeBlanc,R.L.,"Introducing Computer Science Fundamentals Before Programming", ***Frontiers in Education Conference***, Pittsburg, 285-289(1997).

5. Mansoor, A. And Mohammed, H., "Design of an Arabic Programming Language", ***Computer Languages***, 21:191-201(1995).

6. Guibert,N. and Patric,G., "Teaching and Learning Programming with a Programming by Example System", ***International Symposium on End User Development***, Bonn(2003).

7. Cooper,S.,Dann,W. and Pausch,R., "Alice:A 3-D Tool For Introductory Programming Concepts", ***Journal of Computing Sciences in Colleges***,15:107-116(2000).

8. Adamchik,V. and Gunawardena,A., "A Learning Objects Approach to Teaching Programmig", ***International Conference on Information Technology: Computers and Communication***, Las Vegas, 96-99(2003).

EKLER

EK-1 Genel yazım kuralları

1. Yazılacak program Başla komutu ile başlar, Dur komutu ile sonlanır.
2. Yazılan her komuttan sonra mutlaka bir boşluk bırakılmalıdır.
3. Her komut satırının hemen sonuna "**;**" konulmalıdır. Noktalı virgül, komut ve deyimleri birbirinden ayırmak için kullanılır.
4. Programın okunurluğu açısından, her bir satıra bir komut yada deyim yazılır.
5. Diğer programlama dillerinde olduğu gibi değişken tanımlama işlemi yapılmaz.
6. Program yazımında Türk alfabesinin küçük harfleri kullanılır.
7. 0'dan 9'a kadar olan rakamlar kullanılır.

ÖZEL İŞARETLER

Program aynı anda iki adet sayı için matematiksel yada mantıksal işlem gerçekleştirir.

1. + Toplam işlemi için kullanılır.
2. - Çıkarma işlemi için kullanılır.
3. / Bölme işlemi için kullanılır.
4. * Çarpma işlemi için kullanılır.
5. < küçüktür işareti
6. > büyüktür işareti
7. <= küçük eşit işareti
8. >= büyük eşit işareti
9. = eşittir işareti
10. <> Eşit değil işareti
11. := değişkenlere değer atamada kullanılır. Bir atama deyimi gibi iki karakter birlikte kullanılan sembolleri birbirinden ayırmayın. Örneğin A:=B*C

EK-2 Komutların kullanımı

1. **BAŞLA :** Program yazımının başladığını ifade eder.
2. **OKU :** Kullanıcının veri girişini sağlayan komuttur.

Örnek : oku a;

3. **YAZ:** Bilgisayardaki verinin veya istenen bir sonucun görüntülenmesini sağlayan komuttur.

Örnek : yaz c;
Örnek : yaz (a b den küçüktür.);

Yaz komutu ile herhangi bir değişkenin içeriği görüntülenmek istendiğinde, o değişkenin adını yazmak yeterlidir. Ancak yaz komutu ile bir mesaj görüntülenmek isteniyorsa, yapılması gereken mesajın parantezler içerisinde yazılmasıdır. Unutulmaması gereken diğer bir noktada yaz komutundan sonra bir karakter boşluk bırakılması gerekliliğidir.

4. **EĞER…… İSE …….DEĞİLSE :** Mantıksal işlem yapılmasını sağlayan komuttur. Genel kullanımı aşağıdaki gibidir :

EĞER Kontrol ifadesi İSE İfade 1 DEĞİLSE İfade 2

Kontrol ifadesi sonucu doğru ise İfade 1 ile belirtilen komutu çalıştırır. Kontrol ifadesi sonucu yanlış ise de, değilse İfade 2 ile belirtilen komutu çalıştırır. Bazı durumlarda değilse kısmına gerek olmayabilir. Diğer bir deyişle Eğer ve ise kullanımı zorunlu iken değilse kullanımı zorunlu değildir.

Örnek :
eğer a<b ise c:=a+b değilse yaz a;
eğer a<=b ise yaz (a b den küçüktür.) değilse yaz (a b den büyüktür.);
eğer a>=b ise c:=a+b;

EK-2 (Devam) Komutların kullanımı

5. DÖNGÜ : Bir grup komut veya deyimin, belli sayıda işletilmesini sağlayan komuttur. Bir komutun sizin belirleyeceğiniz miktarda sürekli çalışmasının isteneceği durumlarda kullanılır.

Yazım Kuralı :

DÖNGÜ değişken:=Başlangıç değeri,Bitiş Değeri;

Komut(lar)

DÖNGÜSONU değişken;

Değişken, ilk olarak başlangıç değerini alır ve bitiş değerine kadar 1'er artar. Değişken, barındırdığı her bir değer için döngü bloğu içerisindeki ifadeleri bir kez işletir.

Örnek :

```
döngü i=1,4;
c:=a+b;
yaz c;
döngüsonu i;
```

Yukarıdaki örneği incelediğimizde, döngünün başlangıç değerinin 1, bitiş değerinin ise 4 olduğunu görüyoruz. Dolayısıyla bu döngünün, 1'den 4'e kadar toplam 4 kez çalışacağını da belirler. Burada toplam 4 kez çalışacak olan komutlar, döngü i=1,4; ve döngüsonu i; ifadeleri arasında kalan c:=a+b; yaz c; dir. Dikkat edilmesi gereken nokta bu komutların birbirinden bağımsız olarak çalışmayacağıdır. Önce toplama işlemi, ardından da yaz komutu çalışacaktır. Diğer bir deyişle önce 4 kez toplama işlemi yapılacak, ardından da 4 kez yaz komutu çalışacak anlamında değildir.

EK-2 (Devam) Komutların kullanımı

Özetlemek gerekirse toplama işlemi ve yaz komutu döngü bloğunu oluşturmaktadır ve bu döngü bloğu toplam 4 kez çalışacaktır.

6. İKEN : İken döngüsü, genellikle döngü sayısının belli olmadığı durumlarda kullanılır.

Yazım Kuralı :

İKEN Kontrol İfadesi;

Komut(lar)

DÖNGÜSONU ;

Kontrol ifadesi doğru olduğu müddetçe, döngü bloğu işletilir. Eğer döngünün hemen başında bulunan bu kontrol ifadesi şartı sağlamazsa döngü hiç çalışmaz. Bu döngüde, döngü bloğu İKEN ve DÖNGÜSONU komutları arasındaki ifadelerdir.

Örnek :

```
iken a<b;
c:=a+b;
yaz c;
a:=a+1;
döngüsonu ;
```

Yukarıdaki örnek incelendiğinde, döngünün çalışma şartının a'nın, b'den küçük olduğu durumlar olduğu söylenebilir. Diğer bir deyişle a değişkeninin barındırdığı değer, b değişkeninin barındırdığı değerden küçük olduğu sürece döngü çalışır. Döngünün kaç kez çalışacağı ile ilgili bir yorum yapabilmek mümkün değildir.

EK-2 (Devam) Komutların kullanımı

7. TEKRARLA : Tekrarla döngüsü, İken döngüsü gibi, döngü sayısının belli olmadığı durumlarda kullanılır. Döngünün kontrol ifadesi, döngünün başlangıcında değil, sonundadır. Bunun sonucu olarak da TEKRARLA döngüsü her şart altında en az bir kez çalışır. Döngünün çalışma şartı ise, kontrol ifadesi doğru sürece veya kontrol ifadesi yanlış olduğu sürece çalışır denilebilir.

Örnek :

```
tekrarla ;
c:=a+b;
yaz c;
a:=a-1;
olanakadar a<b;
```

Döngü bloğu TEKRARLA ve OLANAKADAR komutları arasındaki ifadelerdir.

8. DUR : Yazılan programın sona erdiğini bildiren komuttur.

EK-3 Programlama ipuçları

Verilen bir problemin anlaşılması çözüme giden yoldaki en önemli adımlardan bir tanesidir. Diğer bir önemli adım ise problemin çözümlenmesidir. Çözümlemeden kastedilense problemde geçen her bir kelimenin komut olarak karşılığının bulunabilmesidir. Eğer bir problem anlaşılmış ve çözümlenmiş ise çözüm için geriye kalan tek şey kodlamadır. Bir problemi çözümleyebilmek için kullanılacak ipuçları aşağıda sıralanmıştır.

Karşılaşılan problemde;

1. Klavyeden girilen, dış dünyadan girilen gibi ifadeler, kullanıcıdan veri girişi istendiğinin göstergesidir. Veri girişi için OKU komutu kullanılır. OKU komutu ile birlikte bir adet değişken kullanılmaktadır. Kullanılan bu değişkenin görevi ise kullanıcının klavyeden gireceği değeri içinde barındırmak diğer bir deyişle saklamaktır.

2. Sonucu yazdıran veya ekrana yazdıran gibi ifadeler YAZ komutunun kullanımını zorunlu kılar. YAZ komutu da bir değişkenle birlikte kullanılır ve değişkenin sakladığı değeri ekrana yazdırır.

3. Matematiksel bir işlemden söz ediliyorsa; yine değişkenler sahneye çıkacak ve içlerinde barındırdıkları değerleri temsilen, işlemi gerçekleştireceklerdir. Yapılan işlemin sonucu ayrı bir değişkenin içine atanacaktır. Dikkat edilmesi gereken bir nokta ise çarpma işlemini temsil eden işaretin " * " olduğudur.

4. Problem, "girilen iki sayıdan büyük olanın" gibi bir karşılaştırma ifadesi içeriyorsa EĞER komutu kullanılır.

5. Problem, "Ekrana 5 kez yazdıran" veya "a sayısı, b sayısından büyük oluncaya kadar yazdıran" ifadesi gibi aynı işlemin birden fazla tekrarını istiyorsa, bu durumda DÖNGÜ, İKEN veya TEKRARLA komutlarından bir tanesi kullanılır.

EK-3 (Devam) Programlama ipuçları

Bu komutlardan hangisinin probleme uygun çözüm olduğunun bulunması için de bazı şartlar mevcuttur. Örneğin, tekrarlanacak işlemin sayısı belli ise “5 kez” DÖNGÜ komutu kullanılmalıdır. “a sayısı, b sayısından büyük oluncaya kadar yazdıran” gibi bir ifadenin varlığında da İKEN veya TEKRARLA komutlarından bir tanesi kullanılabilir. Bu komutların birbirinden farkları ise; İKEN komutunda şart kontrolü tekrarlı yapının başında yapılır. Bunun anlamı, şart sağlanmazsa tekrarlı yapı hiçbir zaman çalışmaz. TEKRARLA komutunda ise şart ifadesi, tekrarlı yapının sonunda kontrol edilir. Bunun anlamı da, tekrarlı yapı her durumda en az bir kez çalışacağıdır.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: ARABACIOĞLU, Taner
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 06.08.1976 Aydın
Medeni hali	: Evli
Telefon	: 0 (256) 227 03 61
e-mail	: tanerarabacioglu@hotmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Gazi Üniversitesi	
	Bilgisayar Eğitimi Bölümü	1999
Lise	Adnan Menderes Anadolu Lisesi	1994

İş Deneyimi

Yıl	Yer	Görev
1999-2001	Milli Eğitim Bakanlığı	Öğretmen
2001-2006	Adnan Menderes Üniversitesi	Öğretim Görevlisi

Yabancı Dil

İngilizce

Hobiler

Futbol, Sinema