Üniversite öncesi öğrencileri için

Türkçe Algoritma Dili Tanımı: *Anlat$_{2005D0}$*

Anlat$_{2005D0}$: A Turkish Algorithmic Language Definition for College Students

.

Bu bildiri 9-11 Kasım Ankara'da Bilgi Teknolojileri Işığında Eğitim: BTIE2005 kongresinde sunulmuş ve bildiri kitabı sayfa 87-97 de yayınlanmıştır.

YAZAR ÖZGEÇMİŞLERİ

M. Ümit Karakaş (Hacettepe Üniv.): Elektrik müh. (ODTÜ 1972), Bilgi Teknolojileri YL (ODTÜ 1973), Doktora (HÜ Enformatik1979), 1992'de Bilgisayar Bilimleri anabilim dalında profesör unvanı aldı. Yapay Us, Yazılım Mühendisliği, Tıbbi Uzman Sistemler ilgi alanlarıdır.

Mustafa Ege (Hacettepe Üniv.): İstatistik lisans (Hacettepe 1984), 1984'de HÜ Bilgisayar Müh. Kadrosuna katıldı. İstatistik YL (Hacettepe 1986), Doktora (Bilgisayar 1993), 1994 yılında Y.Doç. Kadrosuna atandı, halen Veri Yapıları ve Algoritmalar, Algoritma Çözümleme, Çizge Kuramı derslerini vermektedir.

N. Kaya Kılan(Başkent Üniv.) : Matematik-Mekanik (AÜFF 1959), Software Engineering (London College, 1968), İstatistik YL (GAZİ Üniv. 1971), Programming Methodology (München,Informatics Institüde 1973).

Türkiye'de ilk bilgisayarın kuruluşunda görev almış programcı. Birinci kuşak bilgisayardan(1960) buyana; Bilişim alanında: Programcı, Çözümleyici, Araştırmacı, Yönetici ve Bilgisayar Bilimleri dalında Öğretim Görevlisi. Programlama, Programlama Dilleri, Algoritmalar, Bilişim Sistemleri ve Teknolojileri ile Türkçe Bilişim Terimleri ilgi alanıdır.

Ebru Sezer (Hacettepe Üniv.): Bilgisayar mühendisliği (Hacettepe 1996), Bilgisayar Müh. YL (Hacettepe 1999), halen Doktora çalışmaları sürüyor. Veri Yönetimi, Nesneye Yönelik Yazılım geliştirme, Nesneye Yönelik programlamaya giriş ve programlamaya giriş derslerini vermektedir.

Ali Yazıcı (TOBB Ekonomi ve Teknoloji Üniv.): Matematik Lisans (ODTÜ 1972) ve Matematik YL. (ODTÜ 1974) ve Doktora (Waterloo 1983 Bilgisayar Bilimleri) halen profesör ve TOBB Ekonomi ve Teknoloji Üniversitesi, Bilgisayar Mühendisliği Bölüm Başkanıdır. Araştırma konu-ları arasında, Simgesel Hesaplama, Veritabanı Sistemleri, Programlama, e-Öğrenme, Bilişim Etiği ve e-Devlet bulunmaktadır.

0. Özet

Türk lise öğrencilerinin bilgisayar programlama kavramlarına yaklaşırken kolay başlangıç adımları ile başlayabilmesi için on dokuz kolay kurala dayalı, Türkçe söz dizim kurallı bir algoritma dili, Anlat2005D0 tanımlanmıştır. Tanımlanan Algoritma dili, başlangıç düzeyi programlama öğretiminde, daha sonra laboratuarda kullanılacak programlama dilinin ayrıntılarının öğretilmesini ötelemeye yarar sağlamakta, böylece algoritma kurmanın mantığına yoğunlaşma sağlanabilmek-tedir. Bir algoritma diline ve algoritmik yaklaşıma dayalı olarak eğitim/öğretim yapılmasının sadece bilgisayar programı yazabilme yeteneğini geliştirmekle sınırlı olmadığı, öğrencinin belirli (deterministic) düşünme ve bütünleştirerek çözüm (sentez) yeteneklerinin de gelişmesine yardımcı olacağı düşünülmektedir.

1. Niçin liseler için <u>de</u> Türkçeye benzer sözdizimli bir algoritma dili ($Anlat_{2005D0}$) gerekliliğine inanıyoruz...

Son yıllarda lise öğretiminin "küçük boyutlu çözümleme (analiz) alışkanlığını geliştirebildiği fakat "bütünleştire-rek çözüm bulma" niteliğini geliştiremediği çeşitli ortamlarda sıkça konuşulmuştur. Ancak, öğrencinin bütünleştirme yeteneğini geliştirmek için kurulan laboratuar ve ortam sayısı çok sınırlı kalmıştır. Liseler tarafından edinilen bilgisayar laboravuarları da, bilgisayar programlama dersleri için kullanıldığı takdirde (çoğunlukla internette çapraz denetimden geçirilmemiş ham bilgi aramakta kullanılıyor) kısmen bütünleştirme yeteneğini geliştirebilmektedir.

Ancak, derleyicisi (compiler) bulunan programlama dillerinin (liselerde Basic, Pascal, Fortran, C moda) İngilizce olduğunu ve öğreticilerin algoritma ve mantıksal kuruluş anlatmak yerine programlama dilinin sözdizimine (syntax) ve oku (read), yaz (print) komut-larının biçim parçalarına (format letter & format patterns) daha çok yoğunlaştığı düşünülürse çözüm bulma yeteneğini geliştirme amacı yeterince sağlanmamaktadır.

Oysa, eksiksiz bir mantık yapısı kurmak sadece mühendis, matematikçi, istatistikçi olacak öğrenciler için değil gelecekte çeşitli alanlarda çalışacak kişiler için de önemlidir. Bu açıdan üniversite öğrencileri için hazırladığımız Anlat2005D1 ve Anlat2005D2 algoritma dillerinin bir altkümesi olan Anlat2005D0 Türkçeye benzer sözdizim yapılı algoritma dilini, Türk lise öğretmenlerinin kullanımına sunuyoruz.

Bir algoritma dili ile ilk programlama kavramlarını almanın "günün modası olan programlama dili" ile başlamaya göre daha iyi alışkanlıklar sağlayacağını da düşünüyoruz.

Öte yandan algoritma yazmak bilgisayar mühendisliği (computer engineering), yazılım mühendisliği (software engineering), Fen-Edebiyat Fakültesi içinde Bilgisayar Bilimleri gibi alanlarda "tasarım (design)" aşamasının temelidir. Bilgisayar bilimlerinin kuruluş yıllarından başlayarak algoritma kurmanın araçları araştırmacılar tarafından geliştirilmiştir. Bunların ilk örneği Donald Knuth'un tanımladığı MIX dilidir [Knuth1967]. Daha sonra Horowitz ve Sahni [Horowitz1976] Sparks adı verilen sözde kod dilini (pseudo code language) geliştirmişlerdir.

Bir algoritmik dil arayışı modası geçmiş bir yaklaşım değildir. Ticari yönden çok başarılı kitaplardan Deitel & Deitel'in kitaplarında da [Deitel2004] zaman zaman bazı algoritmalar önce sözde kod ile verilmektedir. Son yıllarda ticari başarı yakalıdığı görülen Maureen Sprankle'de [Sprankle2003] bir algoritmik dil (algorithmic langıage) geliştirmiştir.

Türkiye'de de geçmişte algoritmik diller geliştirilmiştir [Karakaş1987]. Bunların Türkiye'de ve Türkiye dışında örnekleri çoğaltılabilir. Bu çalışma ise Ankara'da, Türkçe dilini dersliklerde kullanarak bilgisayar mühendisliği öğretimi yapan üç üniversitenin ortak bir araç geliştirme çabası olarak değerlendirilmelidir.

Algoritma dili tanımlamak için bir grup çalışması gerekli miydi?

Gerek bilgisayar mühendisliği içinde, gerekse bilgisayar mühendisliği bölümü dışında yüzlerce öğretim üyesi/öğretim görevlisi bilgisayar programlama ile ilgili kavramlar öğretmekte ve derslerini anlatırken yardımcı araç olarak bir

algoritma dilini şahsen geliştirmekte-dirler. Hatta denebilir ki aynı öğretim üyesi ders yılının başında kendi düşüncesinde tanımladığı (ancak net biçimde yazmadığı) algoritma dilini dersin ilerleyen haftaları içinde, bazen kendisi de farkında olmaksızın değiştirebilmektedir. Algoritma dili üzerinde bir standart-laştırma çalışması yapılmadığında aynı üniversitenin aynı yarıyılda aynı dersin farklı şubelerinde ders veren ve çalışma odaları aynı koridor üzerinde bulunan öğretim üyelerinin algoritma yazım biçimlerinde önemli farklılık bulunabildiğini gözlemlenmiştir.

2. Lise öğrenciler için Anlat2005D0 tanımı

Algoritma (algorithm) sözcüğünün kökeni M.S.780 yıllarında bugün Özbekistan sınırları içinde kalan Harzem bölgesinin Harzem (Harizm, Khowarizm) kentinde doğan *Türk asıllı matematikçi Musa oğlu Muhammed* M.S. 825 yıllarında yazdığı "*Kitap-ül Fihrist Al-Cebr ve'l Mukabele" (Cebir ve Mukabele Kuralları ile Hesabın Özlü Kitabı)* isimli yapıtına dayanmaktadır. Bu kitapta onlu sayılar, onlu sayılar ile cebirsel işlemler ve birinci, ikinci derece denklem kurma yolu ile problem çözmenin cebirsel kuramını örneklerle anlatmıştır. Böylece "Cebir" altbilimdalının kurucusu olmuş, kitabının Latince çevirilerinde adının yazılışı olan, *Al-Khowarizmi* sözcüğüne bakarak onlu sayılarla cebirsel hesaplamayı öğrenenlere 12-15.yy Avrupa'sında, "*Algorist*", hesaplamaya "*Algorismus"* adı verilmiştir. Kitabındaki bir başka önemli özellik: onun, cebirsel problem çözme yolunu; adım adım, yalın ve veriler üzerindeki işlemlerle tanımlamasıdır. Batıda Al-Khowarizmi sanı ile ün yapan Musa oğlu Muhammed'in Türkçe sanı "*Harzemli"*dir. Harzemli'nin cebir kitabında verdiği "*problem çözüm yolunu tanımlama yöntemi*", 16.yy'da batıda "Algorithmé", "Algorithm", Osmanlı'da "Harezmi Yolu" , günümüzde "*Algoritma"* adını almıştır. Algoritma kavramı, bilgisayar programlama ve bilgisayar bilimlerinin temelini oluşturmuştur bkz, www.baskent.edu.tr/~kkilan).

İzleyen bölümlerde Anlat2005D0'ın her yazım kuralı örneklerle açıklanmıştır. Kurallar ve örnekler için faklı çiçek simgeleri kullanılmıştır.

k01: Algoritma kabul edilmenin beş kuralını, Knuth 1968'de tanımlamıştır.

A) Algoritma sonlu sayıda işlem sonrasında durmalıdır. Algoritma yazan kişi yazdığı algoritmanın sonlu sayıda işlem sonrasında durup durmayacağını mümkün olduğunca iyi sınayarak incelemelidir.

B) Çıkış veri sayısı bir ya da daha çok olmalıdır. Rasgele sayı üreten bir algoritması sıfır veri girişli örnektir. Çıkış verilerinin bir ya da daha çok olması "insan–bilgisayar" etkileşiminin en az koşuludur. "İnsan–bilgisayar" etkileşiminin daha iyi tasarlanması için algoritmanın gerekli her aşamada kullanıcıdan hangi veriyi beklediğinin ve veri türünün belirtilmesi gereklidir.

C) Giriş veri sayısı sıfır ya da daha çok olmalıdır. Bu verinin değer aralığı sınırlı ve kesikli olmalıdır. Algoritmada tanımlanmış değişkenlerin "tanım alanlarıda (validity range)" belirtilmelidir. Örneğin, insanYasi değişkeni için "tanım alanı" 0 ile 120 olarak tanımlanmış ise klavyeden girilen 133 değeri veri girişinde en erken aşamada reddedilmeli ve veri giriş hatasını işaret etmelidir.

D) Algoritmanın her adımı net ve kesin biçimde tanımlanmalıdır. Algoritma adımları yoruma açık olmamalıdır.

E) Algoritma adımları kendi içlerinde etkin olmalıdır.

k02: Algoritma yazmada bilgisayar ortamı tercih edilmelidir.

Anlat2005 dilindeki *Algoritmanın,* programın yazılacağı bütünleşik geliştirme ortamı (Integrated development environment, IDE) üzerinde yazılması tavsiye edilir. Bu yaklaşımın, öğrencinin klavyeye ve fare kullanma alışkanlığını kazandırması, oluşturduğu algoritmayı saklamayı *(save)* öğrenmesi gibi olumlu etkiler vardır.

Anlat2005 Algoritma dili, algılamayı farklılaştıran **koyu yazı,** normal yazı ve *eğik yazı* gibi yazım biçimleri içermediğinden herhangi bir metin düzenleyici program bağımlılığı oluşturmaz. Yazılmış olan algoritma, programa dönüştürülürken bazı satırlar ilgili programlama dilinin tanım komutlarına *(declaration statement),* bazıları uygulanır komutlarına *(executable statement)* dönüştürülebilir. Bazı algoritma satırları ise programın açıklama satırları *(comment lines)* olarak değerlendirilebilir.

Derleyicilerin işlemediği bu açıklama satırları *(comment lines)* aşağıda verilmiştir:

A) Derleyicinin işlemediği tek satır açıklamalar (//) : program yazılırken derleyicinin (compiler) işlemediği, algoritma tasarlayan ve program yazan kişilerin belgeleme amacı ile yazdığı satırlar bir çift ardışık bölü işareti (*// double slash)* ile satır sonu *(newline)* arasında yer alır. Bunlar derleyici tarafından ele alınmazlar ve işlenmezler.

B) Derleyicinin işlemediği uzun açıklamalar (/ */):* program yazılırken derleyicinin işlemediği, algoritma tasarlayan ve program yazan kişilerin belgeleme amacı ile yazdığı uzun açıklamalar bir satırdan fazla ise "/* */" çiftleri arasına hapsedilirler.

C) Açıklama alanlarında Türkçe karakter kullanılabilir, fakat açıklama içinde açıklama (nested comment) olamaz: açıklama satırlarında Türkçe karakterler kullanılabilir, fakat bir /* eşleşen */ ile kapanmadıkça yeniden bir açıklama başlangıcı yapılamaz.

k03: algoritma adımları mantıksal olarak sıralanmalıdır.

A) Algoritma yazılırken adımların birbirine göreli öncelikleri önemlidir. Bu ilişki "adım adım algoritmik Türkçe (aaaTürkçe)" ile birinci düzey çözümlemede ele alınmıştır. Algoritmada Önce, Sonra, DahaSonra ya da Adım01, Adım02, Adım03,...,AdımNN mantıksal sırayı belirlemek için önemlidir.

B) Adım adım indirgeme (stepwise refinement) bir problem çözme yöntemidir ve algoritma geliştirmekte de kullanılır. Öğrenci ilerleyen aşamada (öğrencilerin hazır olduklarını hissettiği aşamada) adım0l, adım02 yerine 01:, 02: biçimindeki sıra *(sequence)* numaralamasını ve bunların adım adım indirgenmesinde *(stepwise refinement)* elde edilen alt adımları da 02.01 : , 02.02 : biçiminde numaralamaya yönelmelidir.

C) aaaTürkçe'nin içerdiği anahtar kelimeler ve simgeler şu biçimde listelenebilir:

Eğer, İse, Değilse, Yinele, oku, yaz, Önce, Sonra, DahaSonra, EnSonra, (,), {, }

k04: Değişken adı (variable name) anlamlı seçilmelidir.

A) Bir programda kullanılan ve değeri/içeriği değişebilen her veri bir değişken adı ile belirtilmelidir. Doğal dilde (örneğin Türkçe, İngilizce) sözcüklerin sonuna takılar eklenebildiği için, ana anlamları kesin olmakla birlikte, görünümleri kesin değildir. Bu açıdan Anlat2005 algoritmasındaki değişken adları her kullanımlarında aynı biçimde ifade edilmelidir.

B) Simgesel değişken adı uzunluğu 20 karakteri geçmemesi önerilir: Anlat2005'de "değişken adı*",* İngiliz Alfabesindeki bir küçük harf ile başlar, sonraki karakterleri İngiliz Alfabesinde harf (büyük/küçük) ya da ondalık tabandaki rakam ile devam eder. Değişkenin simgesel adını belirleyen bu karakter dizisinin *(character string)* 20 karakteri geçmemesi önerilir. Bu biçimde algoritma içinde tanımlanan bir değişken adı ile bilgisayar ana belleğinin "kullanıcı değişkenleri" alanında, içinde değer saklanabilen ve gereğinde içindeki değere erişilebilen bir konum açılır.

C) Değişken adının anlamsal tutarlılığı olmalıdır: Anlat2005'de, değişkenlerin adı ile içeriği arasında anlamsal tutarlılık *(semantic consistency)* gözetilmelidir. Yani bir değişkenin adı insanYasi olarak tanımlanmışsa içerdiği bilgi de bununla tutarlı olmalıdır.

D) İşletim başlarken değişkenin ilk değeri belirsizdir: Anlat2005'de bir "değişken*"* alanının değeri, kullanıcı bir değişkene geçerli bir veri giriş komutu ile dış ortamdan değer aktarıncaya ya da içerdiği tüm değişkenler daha önce geçerli hale getirilmiş bir ifadenin (aritmetiksel, mantıksal ifadenin) sonucunu aktarıncaya kadar belirsizdir.

E) Değişken adı seçiminde bir öneri: Anlat2005'de bir değişken adı birden fazla sözcüğün birleştirilmesi ile anlam kazanıyorsa, birinci sözcüğün harfleri küçük harf, ikinci ve sonraki sözcüklerin ilk karakteri büyük harf diğerleri küçük harf seçilir. Bu biçimleme, yazılım alanında "camelCase" olarak tanınmakta ve birçok programcı tarafından benimsenmektedir. Bizim burada önerdiğimiz alışkanlık Java programcılarının kullandığı biçimdir.

k05: Anlat2005D0'da Tamsayı, Kesirli ve Karakter değişken ve değişmez türleri vardır.

2005 yılında bilgisayarların büyük çoğunluğu 32 ikil (32 bit) ana bellek sözcüğü sağladığından Tamsayı veri türünün değer alanı artı-eksi iki milyar arasındadır. Bu sınırları geçen yerde ve sayının kesir kısmının önemli olduğu yerde Kesirli veri türü kullanılmalıdır.

A) Anlat2005D0'da değişken türleri (variable types) Tamsayı, Kesirli, Karakter tekil türleri ve bunların dizi (array) biçimleridir. Dizi türü veri yapısı günlük hayatımızda sıkça karşılaştığımız bir yapıdır. Örneğin, satranç tahtası 8 x 8 boyutunda iki boyutlu dizi *(two dimensional array)* yapısındadır.

B) Sağlam bir algoritma yazma biçiminde, değişkenin türünün (type of the variable) açıkça yazılması alışkanlık haline getirilmelidir.

(Üniversite öğrencileri için hazırlanmış Anlat2005D1 ve Anlat2005D2 içinde başka veri türleri de verilecektir.)

k06: Anlat2005D0'da Simgesel Değişmezler ve (Symbolic Constants) Değişmezler (Literals) kullanılabilir.

A) SimgeselDeğişmez: Programın işleyişi sırasında değer değiştirmeyen bazı sabit oranlar, katsayılar ve dizi boyları simgesel değişmezlerle ifade edilir. Anlat2005D0'da "SimgeselDeğişmez" adı en çok 6 karakter uzunluktadır ve sadece İngiliz alfabesindeki büyük harfler kullanılabilir. Simgesel değişmezin içeriği programın işleyişi süresince değiştirilemez. İçeriği (content, value) değiştirilebilseydi o zaman değişmez (constant) olmaz değişken olurdu. Matematikdeki değişmezi bu sınıfa girer.

B) Aksine bir tanım yapılmamışsa sayısal simgesel değişmezin değeri 7 ondalık basamak duyarlık (precision) ile hesaplamalara katılır**.**

<u>Simgesel Değişmezler ve Alfabetik Değişmezler</u>

C) Anlat2005D0 içinde "Karakter (character)" tanımı yaptığımızda bu tanım ISO646 (ASCII: American Standard Code for Information Interchange) karakterlerini kastetmektedir. ISO646 (ASCII) olarak adlandırılan bu kod tablosu en eski kod tablosudur ve Türkçe karakterleri kapsamamaktadır. Türkçe karakterleri (ğ, Ğ, ı, İ, ş, Ş) kapsayan iki önemli standard ISO8859 Table 9 (ECMA128 Latin 5, TS5881) ve UNICODE daha sonra Anlat2005D1/Anlat2005D2 içinde ayrıca ele alınacaktır.

D) Tek karakter (single character) tanımlama: bilgisayar belleğinde *tek bir sembolü* (harf, rakam, özel işaret) saklayabilmek için *tek kesme işareti* kullanılır. 'A', 'f' bilgisayar belleğinde ayrı ayrı A ve f karakterlerini saklar,

gereğinde ekranda görüntüleyebilir. Bu biçimde saklanan karakter bir “byte” yer kaplar.

E) Bir Karakter dizisi (character array) ise bir çift, çift kesme işareti (double quote) içine hapsedilmelidir. Bir adet karakteri çift kesme işaretleri arasına yazarsak bu tanım, karakter dizisi sonlayıcı ile birlikte iki “byte” yer gerektirir.

“karenin kenarı =” ve “karenin alanı =“ bilgisayar belleğinde saklanan, gerekli olduğu anda bilgisayar ekranında görüntülenmek üzere gönderilen “karakter dizisi (character array)” örnekleridir.

F) Çift kesme işareti ile sınırları tanımlanmış karakter dizisi içinde boşluk karakteri bulunabilir. Anlat2005D0 içinde diğer özel karakterlerin (örneğin “ işaretinin) karakter dizisi içine konulması ile ilgili ayrıntılar tanımlanmamıştır.

k07: Anlat2005D0’da anahtar sözcükler (key words) ve ayrılmış sözcükler (reserved words) vardır.

Bilgisayar programlama dillerinin çoğunluğunda özel bir anlamı bulunan sözcükler vardır ve bunların değişken ve yordam adı olarak seçilmesi güçlükler yaratabilir. Bazı programlama dillerinde ise bunların değişken ve yordam adı olarak kullanımı yasaktır. Değişken adı olarak kullanımı yasak ise bunlara ayrılmış sözcükler denilir. Özel anlamı olduğu için Anlat2005 içinde değişken adı olarak seçilmemesi gereken sözcükler şunlardır:

Adım01, Adım02, Adım03, Adım04, Adım05, Adım06, Adım07, ..., Aksihalde, ASCII, Bitti, Bloksonu, C-gibi, DahaSonra, Değilse, Dışyapılar, Dizinaltsınırı, Durum, Eğer, Eger, Fortran-gibi, ISO8859, İçyapılar, İse, Karakter, Kesirli, Komutlar, oku, Önce, Pascal-gibi, SayarakYinele, Seçenek, SimgeselDeğismez, SınaDoğruysaYinele, Sonra, Tamsayı, Tanımla, UNICODE, yaz, Yordam, Dön, Döndür.

Türkçe karakterilerin desteklenmediği metin düzenleyicilerde sadece ingiliz alfabesinin harflleri kullanılarak ilgili anahtar kelimelerin yazımı mümkündür.

Yukarıdaki sözcüklerden oku ve yaz dışındakiler zaten büyük harf ile başladıklarından, değişken adı ve yordam adı olmaya uygun değildir.

k08: Algoritma, “insan–bilgisayar” etkileşimini dikkatle oluşturmalı, etkileşim kullanıcıyı tereddütte bırakmayacak bir süreç içinde olmalıdır.

A) oku(...) ve yaz(...) işlevleri: OKU(klavye, değişken listesi) insan ortamından bilgisayar ortamına belirtilen değişkenlerin o işletimdeki değerlerini taşır.

YAZ(ekran, değişken listesi) ise bilgisayar ortamından insan ortamına mesajlar ve değişken değerleri taşır.

Değişken listesindeki değişkenler virgüllerle ayrılır. Algoritma içinde, okuma ve yazma işlemlerinin biçimi tanımlanmamalıdır. Değişkenlerin değerinin ekrana yazılış biçimi, eldeki program açısından çok önemli ise bu biçimler açıklama olarak yazılabilir.

Örnek : KareAlani02.txt

```
Yordam kareAlani ;
Kesirli  kareKenari, kareAlani
Komutlar ;
Adım01 :  oku( klavye, kareKenari )
Adım02 :  kareAlanı = kareKenari * kareKenari
Adım03 :  yaz(ekran, “karenin kenarı =”,
              kareKenari, “  karenin alanı = “, kareAlani )
Adım04 :  Dur ;
kareAlani  Bitti ;
```

Örnek : KareAlani03.txt

```
Yordam kareAlani
Kesirli  kareKenari, kareAlani ;
Komutlar ;
01 : yaz(ekran, “ karenin kenar uzunlugunu cm
                olarak    veriniz ” ) ;
02 : oku( klavye, kareKenari ) ;  //
```

```
03 :  kareAlanı = kareKenari * kareKenari ;
04 : yaz(ekran, kareKenari, kareAlani ) ;
                // kareKenarı   tek ondalık basamak,
                // kareAlani çift ondalık basamak duyarlık
                //  ile yazılmalıdır..
05 : Dur ;
kareAlani  Bitti ;
```

Yukarıda KareAlani02.txt'de karenin alanını hesaplayacak bir algoritma verilmiştir. Bilgisayar modeli D0 çizilip, bu algoritma nedeniyle kullanıcı belleğinde oluşan kareAlani, kareKenari değişkenlerinin değerlerinin oluşumu incelendiğinde mantıksal sıranın doğru olduğu görülmektedir. Yani, önce insan ortamında oluşan kareKenarı bilgisi, adım1 ile bilgisayar belleğindeki kareKenari değişken alanına aktarılmaktadır. Bu oluştuktan sonra bilgisayar belleği içinde kareAlanı değişkeni, değeri doğru olarak oluşmuş olan kareKenari değerinden hesaplanmakta ve üçüncü adımda bilgisayar ortamında hesaplanmış olan değerler insan ortamına verilmektedir.

Ancak, KareAlani03.txt'de verilen algoritma insan– bilgisayar etkileşimini daha iyi tanımlamaktadır.

KareAlani02.txt algoritmasında bilgisayar sessizce, insandan bir veri beklemektedir. KareAlani02.txt algoritmasına dayalı olarak yazılacak bir bilgisayar programının doğru çalışması, bu programın kullanıcısının programın veri beklediğini ve bunun cm cinsinden kareKenari olduğunu bilmek, hatırlamak zorundadır.

Diğer yandan, KareAlani03.txt algoritması sessizce beklememekte, KareAlani03.txt algoritmasına dayalı olarak yazılacak program açılır açılmaz , "karenin kenar uzunlugunu cm olarak veriniz" mesajını, ekrana yazmaktadır.

B) oku(...) komutunda ilk parametre yazılmazsa klavye varsayılır. yaz(...) komutunda ilk parametre yazılmazsa ekran varsayılır. Okuma ya da yazma işlevi bir disk kütüğünden (disc file) yapılacaksa ilk parametre olarak kütük adı ve uzantısı yazılmalıdır**.**

k09: Anlan2005D0'da komut sonlayıcı ayıraç genellikle yeni bir satıra geçişdir. Noktalı virgül kullanmak okunulurluğu kolaylaştırır.

Algoritmanın yazılışında yeni bir satıra geçilmesi ve bu satırda da oku, yaz vb. özel bir anlamı olan bir sözcüğün bulunması komutların birbirinden kolayca ayrılmasını ve algoritmayı okuyan diğer insanlar tarafından tereddütsüz algılanmasını sağlar. Algoritmayı yazan kişi komutların birbirinden kolay ayrılmasını sağlamak için isterse noktalı virgül ";" kullanabilir.

k10: Anlat2005D0'da aritmetik ifadeler tanımlanabilir ve aktarma işleci ile bir değişkene aktarılabilir.

A) Aktarma işleci (ya da =): Anlat2005 Algoritma dilinde aktarma işareti için "**'** işareti tercih edilir. Ancak iki karakterden oluşan bu işarette yaşanabilinecek bir güçlük söz konusu ise "=" işareti de sağ tarafında oluşan hesaplanmış değeri, kendisinin solunda bulunan değişkene aktaran bir "aktarma işareti"dir.

Örnek

```
N = 10        // N değişkeni içeriği tamsayı veri
              //  türünde 10 oldu
N = N + 1     //  şimdi  N  değişkeni içeriği  önceki 10
              //  değerini 1 artırarak 11 oldu
N = N * N     // şimdi  yeni N  değeri içeriği
              //  11  çarpı 11 in eşdeğeri olan 121 oldu
```

Örnek

```
N  10         // N değişkeni içeriği tamsayı veri
              // türünde 10 oldu
N  N + 1      // şimdi  N  değişkeni içeriği  önceki 10
              //  değerini 1 artırarak 11 oldu
N  N * N      // şimdi  yeni N  değeri içeriği
              // 11  çarpı 11 in eşdeğeri olan 121 oldu
```

*B) Aynı parantez düzeyinde, iki işlenenli (two operand) aritmetik işleç (+ - / *) öncelikleri:* Bilgisayar programlama dillerinde yatay bölü çizgisi yoktur. Bu açıdan algoritma yazarken sadece + (artı), - (eksi), / (bölü) ve * (çarpı) işleçlerini kullanabilirsiniz. Aynı parantez düzeyinde / (bölü) ve * (çarpı) işleçleri + (artı), - (eksi) işleçlerine göre önceliklidir.

Aynı parantez düzeyinde bulunan * (çarpı) işleçlerinde öncelik soldaki çarpı işlecindedir.

Örnek:

f = a * a + 2 * a * b + b * b /* önce çarpı işleçleri kullanılarak üç adet değerler oluşturulur. Sonra + işlecinin sıracı geldiği için değerler soldan sağa toplanır */

k (w + 2) * (w – 3) /* burada her iki parantez düzeyi de birinci düzeydedir. w Değişkeninin o andaki değeri alınarak (örneğin 7 olsun) w + 2 ve w – 3 değerleri oluşturulur ve çarpım oluşturulur*/

z = w + 2 * w – 3 /* burada ise ilk yapılan işlem 2 ile w değişkeninin o andaki değerinin çarpımıdır */

Aynı parantez düzeyinde bulunan - (eksi) ve + (artı) işleç sıralamasında öncelik yine soldan başlayarak uygulanır.

Anlat2005 de aynı parantez düzeyindeki / (bölü) işleçlerinde öncelik kuralı belirlenmemiştir. Bu durumda algoritmik düzeyde önceliğin ne olduğu bir parantez çifti ile belirlenmesi gereklidir. Bu tedbir yazılım güvenilirliğini artırmak için alınmıştır.

C) Anlat2005 algoritmasında aynı parantez düzeyinde yan yana bölü işleçleri gelmemelidir. Bu nokta matematikte yatay bölü çizgilerine alışkın kullanıcının hata yapabileceği noktalardan biridir. Yazılım güvenilirliğini artırmaya katkı olmak üzere, Anlat2005 algoritma dilinde birbirine aynı parantez düzeyinde komşu olan bölü işleçlerinden hangisinin önce yapılacağı açıkça, parantez çifti ekleyerek belirtilir.

Örnek

```
//Bölü işleçlerinin olabilecek sıralama biçimleri

z = a / b / c ;   // Anlat dilinde izin verilmeyen bir durum
z = ( a / b ) / c ;   // Anlat dilinde kabul edilen bir durum
z = a / ( b / c ) ;   // Anlat dilinde kabul edilen bir yazım
```

D) Aritmetik ifadelerin okunulurluğunu artırma: aynı parantez düzeyinde aritmetik işleçlerin (+ - / *) yazılması sırasında işlecin öncesinde ve sonrasında birer boşluk karakteri konularak algoritmanın okunurluğu artırılır.

E) Tekil eksi (unary minus): bir değişmeze ya da parantez içine alınmış bir aritmetik ifadeye etki yapan tekil eksi, işleci etki yaptığı gruba bitişik yazılır. Aktarma işlecine komşu olan tekil eksi işlecinin tek bir değişmezi nitelediği durum dışındaki tekil eksi işlecinin nitelediği değişmezler parantez içine alınırlar.

Örnek

```
 beta =  -10 ;
alfa = -( beta * z * z ) ;
yön = - yön ;
döndür = döndür * ( -1 ) ;
```

k11: Anlat2005D0 İşleçleri *(operators)* hazır yordamları (procedures) içerir.

Bir yazılım teorisi çalışması, programdaki herbir birimi ya işlenen *(operand)* ya da işleç (*operator)* olarak ele alır. Bu açıdan algoritma yazmak, "hikaye yazmak" stiline yakın değildir. Algoritma yazmak, yazılım fiziği yaklaşımının felsefesine uygun biçimde işlenenler *(bunların en önemlileri programın değişkenleridir)* ve bunlar üzerine işlemler yapan işleçlerin *(operator)* sorunu çözmek için doğru uygulanışının tanımlanmasıdır.

A) oku(klavye, değişken listesi) ve yaz(ekran, değişken / değişmez listesi) birer fonksiyondur.

```
Kesirli  kediYasi ;
oku( klavye, kediYasi ) ;
```

Bilgisayarın *oku(klavye, kediYasi)* komutunu uygulamaya başlayıp, kullanıcının klavyeden bir değer girmesini beklemeye başladığı anda oku yordamı devreye girer. Örneğin klavyeden 1 tuşuna basıldığında bu önce ISO646 tablosunda bir kod' dur. Bunun sayısal değer 1 haline getirilmesi oku yordamı içinde yapılır. Ancak bu noktada oku yordamı *kediYasi* değişkenine 1 değerini aktarmayı erteler. Kullanıcı 2 tuşuna bastığında daha önce var olan 1 değerinin 10 ile çarpılması ve üzerine 2 eklenerek 12 değerinin oluşturulması da oku yordamında yapılır. Oku yordamı ana bellekteki kediYasi değişkenine değer aktarımını hala bekletiyor durumdadır. Kullanıcı "enter" tuşuna bastığında oluşan 12 değeri oku yordamından bilgisayarın "ana belleğindeki *(main memory)*" *kediYasi* değişkenine aktarılır.

B) Anlat2005'de üs alma işlemi, tamsayı kısmını alma, mod işlemi vb bir simge ile sağlanmaz. Bunlar ve diğer matematiksel işlemler adları İngilizce olarak tanımlanmış yordamlar ile tanımlanır. Başlangıç düzeyi öğrencisinin

alışkanlık kazanması için seçilmiş az sayıda matematiksel yordam örnekten sonra listelenmiştir.

Örnek: DaireAlani.txt

```
SimgeselDeğişmez  PI = 3.141593  ;
Adım01 : oku( klavye, yariCap ) ;
Adım02 :  daireAlani = PI  * pow(yariCap, 2.0 ) ;
          // burada yariçapın karesi alınıyor
Adım03 : yaz(ekran, yariCap , daireAlani ) ;
Adım04 : Dur ;
```

Tablo1: Anlat2005D0'da hazır matematik ve geometrik yordam örnekleri listesi:

Adı	Parametreleri	Ne döndürür
abs(x)	x:değişken	Verilen değişken değerinin mutlak değerini döndürür. b = ABS(-17.3) ; b değişkenine 17.3 aktarır.
imod(x,y)	x:Değişkeny:Bölen	Değişkenin değeri bölen ile bölündükten sonraki kalan(remainder) kısmını verir
real(x)	x:Tamsayı işlenen	Verilen Tamsayı değişkene ya da değişmeze değer olarak eşit fakat bilgisayar belleğinde kesirli (real) bilgi saklama biçimindeki eşdeğerini verir.
rint(x)	x:Kesirli işlenen	Tamsayı oluşturmak için Yuvarlanmış (rounded) değer k = rint(44.4) k içine 44 değerini, k = rint(44.5) k içine 45 değerini getirir
lint(x)	x:Kesirli işlenen	Alt tamsayıyı verir. Yani k = lint(44.4) k içine 44 değerini döndürür
uint(x)	x:Kesirli işlenen	Üst tamsayıyı verir. Yani k = uint(44.4) k içine 45 değerini döndürür
pow(x,y)	x:Taban y:üs	Verilen taban değerinin (kesirli olabilir) belirtilen üs değeri ile işlenmesinden elde edilecek sonucu kesirli değer olarak döndürür. Örneğin a = POW(4.0, 1.5) ; işlemi a değişkenine 8.0 aktarır.
srand(time())	time()	Genellikle srand(time()) biçiminde kullanılıp rasgele bir anda ilk rasgele sayıyı üretir
rand()	-	Genellikle bir srand(time()) çağrılışından sonra tekrarlama aralığı çok uzun olduğu için rasgele dediğimiz, gerçekte "rasgele görünümlü (pseudo random) sayılar üretir
strncmp(x,y,n)	x:karakter katarı, y:karakter katarı n: tamsayı	İki katarın ilk n adet karakter içeriği eşitse 0, x katarı alfabetik büyükse -1 ve y katarı alfabetik büyükse 1 üretir.
time()	-	O andaki bilgisayar saatinin değerini milisaniye duyarlığında verir.

Anlat2005D0 içinde karekök işlevi tanımlanmamıştır ama *pow*(taban, üs) hazır yordamı tanımlanmış olduğundan karekök hesabı *pow*(taban, 0.5) ile gerçekleştirilebilinir.

Öğreticiye Not: Öğrencilere "hazır yordam" kavramını öğretebilmek için çok sınırlı bir altküme burada listelenmiştir.

Öğretici gerek görürse sin, cos, tan gibi birkaç hazır yordamı daha bu listeye ekleyebilir.

k12: Matematikte ve Anlat2005D0 algoritma dilinde işleç önceliği (operator precedence) tablosu vardır.

İşleç öncelikliği *(operator precedence) konusu,* ortaöğretim eğitim müfredatı içinde yer alması sebebi ile genel olarak bilinen bir konudur. Bu alandaki bilgi tabanımızı en iyi biçimde Matematik bilimi oluşturur. Bazı programlama dilleri bir parantez çifti eksik yazmak için matematik tabanı olmayan ek sıradüzen *(additional hierarchy)* tanımlamışlardır. Bizim yorumumuza göre matematik tabanı olmayan, bazı programlama dillerinde kısa yazımlar için oluşturulmuş bu ek sıradüzenler yanılgıya neden olabilmektedir. Ayrıca işleç terimi okuyucuda sadece simgesel işleçleri çağrıştırsada hazır yordamlarda bir ya da daha çok parametre alan ve değer döndüren işleçler gibi ele alınmalıdır. Örneğin + işleci iki tamsayıyı parametre alırsa, toplamı gösteren bir tamsayı değer döndürür. Aynı biçimde, *abs* yordamı parametre olarak bir tamsayı ya da kesirli sayı alır ve aldığı değerin mutlak değerini geri döndürür. Bu benzerliğe dikkat edilirse hem simgesel bir işarete sahip işleçler hem de hazır yordamlar için programcının; sağlaması gereken parametre tür ve sayısını ve geri dönen değerin tür ve anlamını bilmesi yeterlidir. Bu sebeple Anlat2005D0'da hazır yordamlar da işleç önceliği listesine rahatlıkla alınmışlardır.

İşleçler	Açıklama
( )	parantez
yordam	Tablo 1'de önerilen hazır yordam listesi
- !	tekil eksi işleci *(unary minus operator*), mantıksal işleç (*logical operator*) DEĞİL
* /	Çarpı ve bölü işleci
+ -	kesirli artı ve eksi işleci
== != < <= > >=	Mantıksal karşılaştırma işleçleri (eşit, eşit değil, küçük, küçük eşit, büyük, büyük eşit)
&&	Mantıksal işleç (*logical operator*) VE
||	Mantıksal işleç (*logical operator*) YA DA

Öncelik, yukarıdan aşağıya doğrudur.

k13: iki çıkış seçeneği bulunan koşullu uygulamalar Anlat2005D0'da ele alınmıştır.

A) *iki çıkış sağlayan koşullar:*
Eğer (koşul) ise { komutlar }
ya da
Eğer (koşul) ise { komutlar } değilse { komutlar }
ya da
Eğer (koşul) ise tek komut;
komut yapısı ile ele alınır.

Eğer sözcüğü yanındaki ( ) parantez çifti içine bir koşul *(condition)* yazılır. Koşulun yazılışı sırasında büyüktür (>), küçüktür (<), eşittir (== dikkat birbirine bitişik iki adet = işareti), büyük ya da eşittir (>=), küçük ya da eşittir (<=) ve eşit değildir (!=) ilişki işleçleri kullanılabilir.

Kesirli değişkenler arasında eşitlik sınaması yapılmamalıdır. Çünkü virgülden sonra ilerleyen basamaklarda karşılaştırma işlemleri beklenen sonucu üretmeyebilir.

Uzunluğu bir karakterden daha uzun olan karakter türü bilgiler *strncmp*(, ,n) işlevi ile karşılaştırılmalıdır.

Mantıksal karşılaştırma işleçlerinin işlenişi, matematik işleçlerinin işlenişine göre daha az önceliklidir. Aşağıdaki örnekte öncelik sırasını gösterebilmek için parantez kullanılmamıştır.

Örnek : alfa* k+0.1 > beta + 3.0

Parantez çiftleri kullanılmadığı halde örnekteki işlemlerin öncelikleri algoritmik dilde ve programlama dilinde bellidir. Önce alfa ile k arasındaki * işleci, sonra beta ve 3.0 arasındaki + işleci, daha sonra alfa * k'nın ara sonucu ile 1.0 arasındaki + işleci uygulanır. En sonunda da daha düşük öncelikli olan büyüktür (>) mantıksal karşılaştırma işlecinin o anda geçerli beta, alfa, k değerleri ile doğrumu (true), yoksa yanlışmı (false) olduğu belirlenir.

B) Mantıksal karşılaştırma işleçleri ve öncelikleri: Anlat2005'de > < >= =< != ve eşit (==) mantıksal karşılaştırma işleçleri Matematik bilimindeki tanımlarına uygun olarak eşit öncelikle tanımlanmıştır. C dilinin matematik tanımdan farklılaşan yanılgısı tekrarlanmamıştır. Öteki deyişle Anlat2005 algoritma dilinde

Eğer (beta > 5.0 == alfa <= 7.0)

İse {..... }

İfadesi geçerli bir koşul yapısı değildir. Yukarıda yazılmış koşul bazı programlama dillerinde geçerlidir, bazı programlama dillerinde geçerli değildir. Yazılım güvenilirliğine katkı yapmak için Anlat2005'de mantıksal karşılaştırma işleçleri eşit öncelikte tanımlanmıştır. Bunun kullanıcıya getirdiği yük yukarıdaki ifadede öncelikleri nitelemek için parantez çiftlerinin kullanılmasıdır.

Anlat2005'deki doğru kullanım örneği aşağıda verilmiştir.

Eğer ((beta > 5.0) == (alfa <= 7.0))

İse {..... }

Algoritmanın biraz daha belirleyici biçimde yazılmasını zorunlu hale getiren bu küçük özellik geliştirilen ve belgelenen algoritmanın çeşitli programlama dillerine daha güvenilir biçimde aktarılmasını sağlar.

C) Anlat2005D0'da değeri doğru (true) ya da yanlış (false) olabilen mantıksal (logical) veri türü sadece koşul yapısı içinde geçerlidir. Anlat2005D0 içinde mantıksal değişken türleri dış ortamdan (insan ortamı) okunmazlar ve insan ortamına YAZ komutu ile gönderilmezler. Programın değişken türleri içinde Anlat2005D0 da açıkça tanımlanan mantıksal değişken türü bulunmamaktadır.

D) DEĞİL , VE, YA DA anlamındaki mantıksal işleçler, mantıksal karşılaştırma işleçlerinden daha düşük önceliklerle tanımlanmıştır. Tekil değil işleci sadece bir parantez çiftinin sol tarafında yer alabilir . DEĞİL işleci, VE ve YA DA işleçleri ile aynı parantez düzeyinde kullanılamaz.

E) Koşul komutunun kısa yazılışı: sadece olumlu koşul sağlandığında tek bir işlem yapılması gereken koşul komutları bir çift kıvrık parantez içine hapsedilmiş komutlar dizisi (blok komut) yerine, koşul komutunun bittiğini işaret eden bir noktalı virgül kullanılarak

Eğer (koşul) İse komut;

biçiminde sonuna bir noktalı virgül eklenilerek fakat kıvrık parantezler arası blok komut oluşturmaksızın yazılabilir. Bu yazımın tek satıra sığabilecek basit yazımlarda kullanılması, diğer durumlarda blok komut yapısının { } tercih edilmesi önerilir.

k14: çok sayıda çıkış seçeneği bulunan koşullu uygulamalar farklı bir yapı ile ele alınmıştır.

Bir noktadan dallanan birden fazla koşulu yönetebilen bu komutun çatısı şöyledir:

```
Seçenek ( tamsayı değişken )
{  Durum 1:  komut grubu 1; Bloksonu ;
   Durum 2:  komut grubu 2; Bloksonu ;

    Durum N:  komut grubu N; Bloksonu ;
   Aksihalde:  komut grubu N+1 ;
}
```

Seçenek yapısı tamsayı değişken yerine tek karakter içeren değişkende olabilir.

k15: Yinelemenin bir sayı kadar gerçekleşmesi gerekiyorsa SayarakYinele (...) {.....} döngü yapısı kullanılır.

Bir sayaç değerinin (döngü denetim değişkeni) başlayış, değişim ve devam koşullarını denetleyerek bir komut grubunu belirli sayıda yineleyen (tekrarlayan) komut yapısıdır. Genel yapısı şöyledir:

SayarakYinele (başlangıç aktarması, devam koşulu, değişim aktarması)
{ yinelenecek komutlar }

Örnek k12-1 : faktöriyel değerlerini 7 ! kadar hesaplayalım

```
Yordam  faktorialBul ;
Tamsayı  nfact, n ;
Komutlar ;
nfact   1
SayarakYinele ( n  1,  n <= 7, n  n + 1 )
// döngü kontrol grubu
{   nfact  n * nfact ;
     yaz( ekran,  " n = ", n,  " faktöriyeli = ", nfact )
                                  //  satır atlayarak yaz
}
Dur ;
faktorialBul Bitti ;
```

k16: Yineleme bir koşul doğru olduğu sürece gerçekleşecekse Önce (.....) SınaDoğruysaYinele { .....} döngü yapısı kullanılır.

Önce sözcüğü yanındaki parantez içindeki koşulun doğru olduğu sürece uygulanacak komut grubunu belirtir.
Burada dikkat edilmesi gereken nokta *Önce* sözcüğünün yanındaki koşul içindeki değişkenlerin değerlerinin bu noktaya ulaşmadan önce belirli (defined) hale gelmiş olmasıdır.

Örnek k12-2 : faktöriyel değerlerini 7 ! kadar hesaplayalım

```
Yordam  faktorialBul ;
Tamsayı  nfact, n ;
Komutlar ;
nfact   1 ;
n  1 ;
Önce (  n <= 7 ) SinaDogruysaYinele
            // döngü kontrol grubu
{    nfact  n * nfact ;
     yaz( ekran,  " n = ", n,  " faktöriyeli = ", nfact )
            //  satır atlayarak yaz
     n  n + 1 ;
}
Dur ;
faktorialBul Bitti ;
```

Dikkat edilmesi gereken diğer nokta ise koşul içindeki değişkenin ya da değişkenlerden birinin, bir süre sonra *(sonlu zaman içinde)* döngüden çıkışı sağlayacak yönde değişim göstermesidir.

k17: *Anlat2005D0* sayısal türden (Tamsayi, Kesirli, vb) tek boyutlu dizileri *(vectors, single dimensioned arrays) destekler.*

A) Anlat algoritma dilinde indis ayıracı köşeli parantezdir: bir algoritma dili olan Anlat tanımlarında indis ayıracı C grubu dillerde olduğu gibi köşeli parantez olarak seçilmiştir.

B) Anlat Algoritma dilinde "boyut alt sınırı" tanımlanmalıdır. Bu esneklik Anlat2005 algoritma dilinin daha sonra kullanılacak olan programlama dili ile daha iyi uyumlaşmasını sağlar. Bu biçimde tanımlandığında bu üç yaklaşımın üçüne de uyarlanabilir. Bunu sağlamak için algoritma gruplarının başlangıcına bir defa #Tanımla komutu konulmalıdır. Örnek,

#Tanımla DizinAltSınırı = {C-gibi, Fortran-gibi, Pascal-gibi }

dizi veri yapısı kullanan her Anlat2005 algoritma grubunda #Tanımla DizinAltSınırı bir kez kullanılmış olmalıdır.

B) Anlat2005D0 da "boyut üst sınırı", bir simgesel değişmez ile algoritma yazılırken tanımlanmalıdır: simgesel değişmez sözkonusu algoritmada, algoritmanın yazıldığı sırada bilinen bir sabiti (constant) ya da bir dizinin o algoritmadaki üst sınırını belirtir. Simgesel değişmez adları İngiliz alfabesindeki büyük harfler ile kurulur ve simgesel değişmez adı en az bir büyük harf en çok 6 büyük harften oluşur. Anlat2005 Düzey-Sıfır'da dizi boylarının simgesel değişmez ile verilmesi gerekir.

k18: Anlat2005D0 ile iki ve daha çok boyutlu diziler tanımlanabilir.

Birbirine paralel diziler üst sınır değeri için aynı simgesel değişmezi kullanılarak yazılır. İki boyutlu biz dizi indislerin iki ayrı boyut olarak köşeli parantezler içinde verilmesi ile yapılır. Örneğin, *tamsayi satrancSahasi[8][8].* Bu komut satranç tahtasındaki taşların kodlarını saklayacak 8x8'lik bir matris (iki boyutlu dizi) yapısı açar.

k19: Yordam tanımı:

Bir yordam tanımı, yordamın adı, yordamın parametreleri, yordamın dış veri/iletişim yapıları, yordamın iç veri yapıları, yordamın komutları ve dön (ya da döndür) komutundan oluşur. Yapıyı oluşturan bölümler aşağıda açıklanmıştır.

```
Yordam VergiHesapla ;
Dışyapilar ;
        /*Burada bu yordamın dışından hangi bilgiler gelecek ise tanımlanmalıdır. Bunların bir kısmı programlama
```

```
    sırasında yordamlar arası parametre iletişimi ile sağlanacaktır. Algoritmanın doğru kuruluşunu tartıştığımız bu
    aşamada, burada dışyapılar bölümünde tarif edilmiş, tanımlanmış olması, daha sonra program yazılırken
    unutulmaması için gereklidir. */
İçyapılar ;
   /*  burada sadece bu yordam içinde kullanılacak
       değişkenler tanımlanır  */
Komutlar;
{

  Dön ;  //  yazılmazsa komutlar sözcüğüne bitişik sol
         // kıvrık parantezin eşi olan sağ kıvrık
         // paranteze ulaşılmış olması
         // çağıran programa dönüşü sağlar
}
VergiHesapla Bitti ;
```

19.1. Yordam adı: Anlat2005 alışkanlığında kullanıcıların yazdıkları yordam adı küçük harf ile başlamalıdır. Yordam adını oluşturan diğer sözcüklerin ilk harfi büyük, diğer harfleri küçük harf olmalıdır. Yordam adını oluşturan karakterler İngiliz alfabesindeki harfler olabilir. Anlat algoritma dilinde pow( ), abs( ) gibi İngilizce adlandırılmış az sayıda "hazır yordam" da bulunmaktadır.

Önemli öneri: Kolay anlaşılırlık açısından yordam adı yordamın yaptığı işi tanımlayan bir fiil, bir emir olmalıdır (yani yordam adında *ortalama* değil *ortalamaBul* tercih edilmelidir)

19.2. Yordamın bilgisayar ortamında diğer program bölümleri ile veri ve kontrol iletişimi Anlat2005D1'de daha ayrıntılı açıklanacaktır. Ön bilgi olarak burada şu bilgi verilebilir:

Bir yordam, diğer yordamlar ile üç biçimde iletişim yapabilir:

* yordamın parametrelerine çağırma anında gönderilen (değer ile çağırma) ya da bağlanan (kaynak ile çağırma) değerler / değişkenler
* dışyapı (external, common, yordam hierarşisi nedeniyle ulaşılabilen) değişkenler
* yordamın kendi adı, içinde bir tek değeri çağıran programa döndürülebilir. a = abs(x) gibi...

Bu iletişim yöntemlerinin anlaşılması Anlat2005D1 kapsamındadır. Anlat2005D0 öğrencisi, bir verinin, üzerinde çalıştığı yordamın iç veri yapısımı olduğunu ya da dış ortamdan gelmesi/dış ortama gitmesi gerektiğinin ayırımına varmalı ve algoritmasında bunu tanımlamalıdır.

19.3. Yordam tanımında içyapılar: Bu noktada öğrencinin kavraması gereken nokta çağıran programda tanımlanmış sayac adlı bir değişken ile çağrılan yordamın içyapısı olarak tanımlanmış, adı *sayac* olan değişkeninin ayrı bellek alanı ve ayrı kapsamlar *(different scope)* olduğunu anlamasıdır. Yani içyapı olarak tanımlanmış bir değişken ya da işlev *(private function)* sadece tanımlandığı yordam içinde bilinir, yordamın dışında ne ad olarak bilinir, ne de değeri taşınabilir.

19.4. Yordam tanımında komutlar: yordamın tanımı nedeniyle yapması gereken işleri yapan kısım bu kesimdir. Bir yordam tasarlanıp kullanıcıya serbest biçimde kullanması için açıldığında yordamın yüzlerce kullanıcısının her birinin yordamın sınırlarını ezbere bilmesi beklenemez. Bu açıdan örneğin sin(aci) yordamına radyan birimine dönüştürülmemiş bir açı değeri geldiğinde

*aci = aci*3.141593 /180;* dönüşümünü yapmak ve kullanıcıya bir uyarı mesajı yazdırmak öğrenciler tarafından kullanılan bir derleyicide uygun bir yaklaşım olabilir. Buna karşılık uzay ve havacılıkta, tıbbi sistemlerde kullanılacak derleyicilerin altyapısında yer alacak hazır yordamlarda "uyarı mesajı (warning)" vermek yeterli değildir. "Hata mesajı (error message)" vermek daha doğrudur.

3. Bir Anlat2005D0 Örneği

Örnek : Problem:
/* sınıf mevcutları 100 öğrenciyi aşmayan sınıflar için öğrenci notlarının bilgisayara girilmesi, not ortalamasının bulunması ve not ortalamasının üzerinde not alan öğrenci sayımı */

Birinci düzey çözümleme : doğal dil (aaaTürkçe : adım adım algoritmik Türkçe)

Önce öğrenci no larını ve notları bilgisayara gir,
Sonra sınıf not ortalamasını bul,
DahaSonra not ortalaması üzerinde not alan öğrencileri yazdır, bu arada say,
EnSonra ortalamanın üstünde not alanların sayısını yazdır.

İkinci düzey çözümleme : Anlat2005D0 algoritma dili ile

```
Yordam ortalamaUstuBul
   //  yazar Ali Yazıcı
Dışyapılar
   // ekran ve klavye iletişimi kurulmalıdır

İçyapılar
#Tanımla DizinAltSınırı =  C-gibi ;
   // bu algoritma  ilk indisi sıfır ile başlayan dizi
   // ( base zero array  system )   için tasarlanmıştır
#Tanımla NMAX  =  100 ;
Tamsayı  ogrenciNo[NMAX ] , say ;
Kesirli   ogrenciNotu[NMAX ] , toplam , ortalama  ;
Tamsayı  n ;
   //  bu sınıftaki öğrenci sayısını tutacak değişken

Komutlar ;
{
yaz( ekran, " ogrenci numarasi ve ogrenci notlarini" ) ;
yaz( ekran, "  bilgisayara giren program başlıyor.  " ) ;
yaz( ekran, "bu program sınıfta öğrenci sayısını " ) ;
yaz( ekran, "ençok 100 alabilir, kaç ogrenci var ? " ) ;
oku ( klavye, n ) ;
Eğer ( n < 1 || n > NMAX )
İse{ yaz( ekran, " programın yeteneklerini aşıyor " ) ;
       Dur ;
     }

SayarakYinele ( k = 1, k <= n, k = k + 1 )
{   yaz( ekran, k, " .inci öğrencinin önce numarasını
                    sonra notunu giriniz " ) ;
    oku( klavye, ogrenciNo[k – 1] , ogrenciNotu[k - 1] ) ;
}

toplam = 0 ;
SayarakYinele ( k = 1, k <= n, k = k + 1 )
{    toplam = toplam +  ogrenciNotu[ k - 1 ]  ;
}
ortalama = toplam / real( n ) ;
yaz( ekran, " ogrencilerin not ortalaması = ", ortalama ) ;
                    // iki satır atlayarak yaz

yaz( , " not ortalaması uzerinde not alan ogrenciler  " );
say = 0 ;
SayarakYinele ( k = 1, k <= n, k = k + 1 )
{   Eğer (ogrenciNotu[ k - 1 ]  > ortalama )
       İse { say = say +  1;
             yaz( ,ogrenciNo[ k - 1] , ogrenciNotu[k - 1]  );
                    //  ortalamanın üzerindeki her öğrenciyi
                    //  yazınca satır atla
           }
}
yaz( , "  not ortalaması uzerinde not alan ogrenci sayisi
              =  " , say ) ;
}
ortalamaUstuBul Bitti ;
```

4. Sonuç ve Öneriler

Burada lise öğrencileri için önerilen Türkçe tabanlı bir sözdizimi olan algoritma dili, ondokuz adet kural örneğimiz incelendikten sonra görüleceği gibi, üniversitelerde çeşitli bölümlere verilen birinci programlama dili kavramlarının %80'ini ve algoritma kavramlarının %100 ünü karşılamaktadır.

Bu noktada Anlat2005D0 içinde sapma komutu *(goto xxx)* ve komut etiketi *(statement label)* bulunmadığına, değer ile çağırma/kaynak ile çağırma konularına girilmediğine dikkat çekmek isteriz. Burada tanımlanmış olan algoritma dilinin, lisede ve üniversite birinci sınıf düzeyinde öğretilen programlama dili bilgilerinin %20' sini dışarıda bırakması zaten amaçlanmış bir durumdur, çünkü burada ilgili programlama dilinin giriş/çıkış komutlarının biçim (format) bilgileri vardır.

Burada önerdiğimiz yaklaşım *(algoritma dili ile başlamak)* tercih edilmeyip "bir programlama dili" ile başlanıldığında gerek öğretici, gerekse bu ders için ders çalışmaya başlayan öğrenci, asıl odaklanması gereken algoritma kurma noktasını bırakıp oku/yaz komutlarının biçim (format) gruplarına odaklanmaktadır. Bunun da hem öğrenci açısından hem öğretici açısından kısa dönemde pratik – uzun dönemde zararlı etkileri vardır: Programlama dilinin sözdizim kurallarını ezberleyerek, oku yaz komutlarının biçim (format) gruplarını ezberleyerek o ders yarıyılındaki sınav sorularının yarısını çözmek ve orta not ile sınıf geçmek mümkündür. Bu durum ise ilk programlama dersini geçmiş fakat algoritma kuramayan öğrencileri programlamada ikinci derse kayıtlı hale getirmektedir. Oysa, burada, algoritma dilinin azaltılarak 19 algoritmik kurala indirgenmiş komutlarının öğrenilmiş olması öğretici (öğretmen, öğretim görevlisi) tarafından ders değerlendirmesinin %8-%10'una kolayca indirilebilir. Bu durumda değerlendirmenin %90'dan fazlası "algoritma kurma" noktasına yoğunlaşacak ve değerlendirme de bu çerçevede yapılacaktır.

Öte yandan, algoritma dili içine konulmuş *#Tanımla DizinAltSınırı = {C-gibi, Fortran-gibi, Pascal-gibi}* komutu ile birçok programlama dili için, sonra uygulama sahasında bozulması gerekmeyen algoritmalar üretmek mümkündür. Öğrencilerin başlar başlamaz değişken adları tanımlamakta karşı karşıya getirildiği kurallar (örnekte *camelCase*), yordam adlarının bir fiil (*ortalama* yerine *ortalamaBul* tavsiyesi) ondokuz algoritmik maddenin tasarımı.., yazarların üniversite öğretim/eğitiminde çağdaş yazına (literatüre) dayalı olarak tercih ettiğimiz güncel yöntemlerden süzülmüştür.

Anlat2005D0 için derleyici (compiler) gerçekleştirme amaçlarımız içinde değildir, ancak biçimleyici (prety printer) ve yorumlayıcı (interpreter) geliştirme çalışmaları bir alt grup ile başlamaktadır.

5. Kaynaklar

Knuth1968 ; *Donald Knuth; the art of computer programming, vol 1 : fundamental Algorithms, Addison wesley, 1968*

Horowitz1976; *Ellis Horowitz & Sartaj Sahni ; fundamentals of data structures; computer science press 1976*

Karakaş1987; *bilgisayar yazılımında veri yapıları ve algoritmalar, sanem matb. 1987*

Deitel2004; *H.M. Deitel & P.J. Deitel ; C how to program, Pearson, fourth editon 2004, ISBN 0-13-122543-X*

Sprankle2003; *Maureen Sprankle ; problem solving & programming concepts; Pearson, 6.basım, 2003, ISBN 0-13-122807-2*

6. Ek1 : Bir derleyici dersi ders projesi olarak Anlat2005D0

Bu aşamada Anlat2005D0'ı anlayan ve yorumlayan (interpret) bir yazılım henüz yok ise de yüksek lisans düzeyinde derleyici gerçekleştirimi (compiler const-ruction) dersi almakta olan bir öğrenci ders projesi olarak Anlat2005D0 yorumlayıcısı üzerinde çalışmaya başlamıştır. Bu çalışmada Anlat2005 dilini beyaz tahtada anlatım sırasında kısmen farklı biçimde anlatabilecek öğretim üyelerinin "Eger" ve "Eğer" gibi kısmen farklı yazımlarına da izin veren esnekliğin derleyicinin tarayıcı (lexical analysis) bölümünde 45 anahtar sözcük (key words, bunlar aynı zamanda ayrılmış sözcük/reserved word), artı 11 anahtar sözcük (bunlar hazır yazılım kavramını öğretmek için seçilmiş çok az sayıda hazır yordam adı) ve 23 adet işleç (operator, bunlar + - * / < <= == => > ( ) [ ] { } gibi tek ya da çift simgelik işaretlerdi) bulunmaktadır. Daha önceki deneyimimiz bu düzeydeki bir yorumlayıcı projesinin iki derleyici dersi öğrencisinin, ya da kod üretme de dahil olmak üzere derleyici (compiler) olarak geliştirmek üzere üç derleyici gerçekleştirimi ders öğrencisinin projesi olabileceğini göstermektedir.
işletim sistemi (windows XP, linux,..), Derleyici (compiler, C, C++..)

7. Ek2 :