Etkili Program Yazma Yöntemleri

Ali Filiz[1], Korhan Günel[1], Taner Arabacıoğlu[2]
[1] Adnan Menderes Üniversitesi, Matematik Bölümü 09010 Aydın
[2] Adnan Menderes Üniversitesi, Aydın MYO 09010 Aydın
afiliz@adu.edu.tr, kgunel@adu.edu.tr, tarabacioğlu@adu.edu.tr

Özet: Bu makalenin amacı, geçerli programlama dillerinden birini (C++, C++ Builder, Pascal, Delphi, LISP, MATLAB, ALGOL, MATHEMATICA, FORTRAN, BASIC, vs.) kullanarak etkili bilgisayar programları yazmak için, dikkat edilmesi gereken hususlar üzerinde çeşitli örnekler verilecektir.

Abstract: The purpose of this article is to present tricks for efficient programming using one of the available computer languages (e.g. C++, C++ Builder, Pascal, Delphi, LISP, MATLAB, ALGOL, MATHEMATICA, FORTRAN, BASIC, etc.). In addition some different examples are given in the last section.

Anahtar Kelimeler: Program, Programlama, Programlama Dilleri, Programcı, Derleyici (Compiler), Cursor, Debug, MATLAB.

1. Giriş

Bilgisayardan yapılması istenen ve belirli bir amaca yönelik olan bir dizi işleme, program adı verilir. Program yazmada sonuca gitmek için basamak basamak tüm adımları sırasıyla gitmek gerekir [7].

Çoğunuz, Internet üzerinden arama motorları ile elinizdeki ödeve uygun program veya programlar aramışsınızdır (Buna bir çok kişi dahildir). Peşinen söyleyeyim, boşuna yorulmayın; bulamazsınız. Bulsanız bile bulduğunuz programı anlayamazsınız. Programı yazan kişi niye bunu böyle yazmış diye düşünür durursunuz. Bunun nedeni programcı programına kendisinden çok şeyler katar. O anda aklından geçen bilgileri irdeler, yorumlar ve kağıt üzerine aktarır ancak yazdığı programa çoğu zaman tam olarak açıklama satırı "comment" eklemez. Bazen, ne kadar "comment" yazarsa yazsın programı inceleyen kişi için mutlaka anlaşılmadık bir kod parçası veya kısımları olacaktır. Bir kere programı anlamak için kişinin programı içeren konu hakkında yeterli ve en ince detaylı bilgiye sahip olması gerekmektedir. Üniversitede Nümerik Analiz ve programlama ile ilgili derslerde hazır yazılmış kaynak kodlu programları araştırma görevlilerine veya öğrencilere hazır verildiğinde; programlar hakkında tam bilgiye sahip olmadıkları gözlenmektedir. Programlar üzerinde bazı küçük değişikliklerde veya yanlışlıkla yaptıkları hata sonucunda programın çalışmama veya hatılı sonuç verme durumunda, öğrencilerin programları düzeltemediklerini kolaylıkla gözlemleyebilmekteyim.

Programcı, eline bir yazacağı program hedefi aldığı zaman ilk önce nereden başlayacağını iyi belirlemelidir. Eğer bunu yapabiliyorsa işin geri kalan kısmı çorap söküğü gibi gelecektir. Burada yöntemi belirlemeli demiyorum, çünkü yöntem programı isteyenler tarafından verilecektir. En önemli ve altın bir kural: Asla program yazmaya bilgisayar karşısında başlamayın! Daima kağıt ve kalem kullanın! Tabii ki yazdığınızı derlerken, "compile" ederken hata çıkacaktır. Bir defada program çalıştıran programcıya çok nadir rastlanılır. Yazdıklarınızı parça parça "compile" edin ki hatanızı daha iyi göresiniz ve sonuca daha çabuk ulaşasınız.

Eğer hâlâ yapamıyorum olmuyor diyorsanız, işe bir alt kurdan başlayın! "Cursor takip metodu"ndan! (siyah ekrandaki "cursor"u takip edin onu oynatın biraz, göreceksiniz çok zevklidir!). Eften püften programlarda (döngülerde **cout, cin** içerikli) biraz şekil oluşturun! Debug neler yapar öğrenin! Program yazarken düşüncelerinizi programa vermenize ve ne yazarsam ne olur gibi kavramları geliştirmenize biraz yardımcı olur! Zaten çoğu kitabın ilk konularında * işaretinden şekiller yaptırılmaya çalışılmıyor mu? Bunlar önemsiz gibi görülebilir ama işin temelinde beyin jimnastiği yatmaktadır.

Eğer oturup alıştırma yapmazsanız ve hazıra konmak istiyorsanız daha çok aşındırırsınız arama motorlarını!

Yazılmış programları incelemekte fayda olduğuna inanıyorum. Bu sayede ufkumuzun genişleyeceğini düşünüyorum (Linux'un Windows'u sollaması bu yüzden değil mi). Yeni başlayan arkadaşlar için olay aynen yukarıda değindiğim gibi. Yoksa temel olmadan program üzerine bir şeyler kurmanın ve program yazmanın bir anlamı olmuyor. Bir de olaya şu açıdan bakabiliriz: Programlama artık bizim işimiz olacak. Derslerden geçmek ile bundan sonra programlamadan kurtulacağımız anlamına gelmiyor. Bu bizim artık hayatımızın bir parçası ve geri kalan bölümünde ya başımızı ağrıtacak ya da karnımızı doyuracak. Öyleyse olaya biraz daha ciddi yaklaşmamız gerekiyor [1].

Çölkesen [3] kitabında:

"Program tasarımı bir sanattır; diğer sanat kolları gibi belirli bir temel eğitim alındıktan sonra bireyin ilgi ve deneyimini uyguladığı, isteklerini ve duygularını yansıttığı ve bir ürünün ortaya çıktığı bir eylem, bir çalışma şekli, bir yaşam biçimidir. Dolayısıyla program tasarımı yapacak bir bireyin her şeyden önce ilgili sanat dalına ait temel kavramları ve bilgileri öğrenmesi gerekmektedir; kendisini gerekli alt yapı ile donatmalıdır. Ancak iyi bir alt yapı ile evrensel sanat yakalanabilir. Yazılım konusu programlamanın üst kümesidir; dolayısıyla yazılım, bilgisayar veya benzeri sayısal cihazlar üzerinde istenen bir işi yerine getirecek ve herhangi programlama dili/ program geliştirme aracı kullanarak işi/görevi yerine getirmek için gerekli program ve verilerden oluşur." diye ifade etmektedir.

2. Programlamadaki Sırlar

Bir bilgisayar programı hazırlanırken aşağıdaki işlemlerin sırasıyla yapılması gerekmektedir.

I. Problemin analizinin yapılması (ANALİZ):

Bir bilgisayar programında temel olarak üç bölüm bulunur. Bunlardan birincisi, işlemin yapılabilmesi için gerekli verilerin (ham bilgi) okunması (GİRİŞ); ikincisi, verilerin üzerinde çeşitli hesaplama, karşılaştırma işlemlerinin yapılması (İŞLEM) ve üçüncüsü ise üretilen sonucun istenen biçimde yazdırılması (ÇIKIŞ) olayıdır. Bu aşamada;

1. Hangi verilerin hangi birimden (disket, ekran, vb) ve nasıl okunacağının,
2. Hangi işlemlerinin hangi sırada yazılacağının,
3. Hangi sonuçların, ne şekilde yazdırılacağının, belirlenmesi gerekir.

Bazen program yazarken analiz bölümü ya tamamen atlanır veya kısa geçiştirilir. Bu, programcıya zaman kazandırmaz, tam aksisine zaman kaybettirir. Çünkü bu bölümde düşünemediği veya gözden kaçırılan bir durum, bir işlemi bir konu ileride ortaya¸ çıktığında programda hata olabilir, yanlış sonuç elde edebilir [7].

II. Algoritmanın yazılması (ALGORİTMA):

Bu bölümde, problemi meydana getiren parçalar tespit edilir. Ortaya çıkabilecek ihtimaller belirlenir, problem serbest yazıyla kaleme alınır.

III. Akış diyagramının (şemasının) çizilmesi (AKIŞ ŞEMASI):

Akış şeması algoritmanın şekil ve semboller ile ifadesidir. Yazılacak programın daha kolay anlaşılabilmesini sağlayan akış şeması, yapılan mantık hatalarının da kolaylıkla görülmesini sağlar.

VI. Seçilen bir dilde (Örneğin MATLAB) programının yazılması (PROGRAM):

Analizi yaptıktan, algoritmayı yazdıktan ve elle adım adım çalıştırdıktan sonra akış şeması çizilir ve son olarak program yazılır.

"Bilgisayar çalışmaya başladığında, programda tanımlanan işlemleri sırasıyla, hızla yerine getirir.

Çoğu insan, bilgisayarların düşündüğünü yani zekası olduğunu zanneder. Henüz tam anlamıyla düşünebilen, zekası olan, kendi kendine iş yapan bir makine geliştirilmemiştir. Bir bilgisayarın hangi işlemleri hangi sırada yapacağına ilişkin tüm bilgiler, bilgisayara programcı tarafından aktarılır. Bilgisayar bu işlemleri program denetiminde milyonlarca işlem hızında yerine getirmektedir. Bu nedenle programdaki işlemler yanlış tanımlanmışsa, ya da programın kullandığı bilgiler hatalı ise bilgisayar bunların hatalı olduğuna bakmaksızın hızla uygulayacak ve bunun sonunda bilgisayardan çıkacak sonuçlar da yanlış olacaktır.

Bir düşünme işlemini insan beyni büyük bir hızla yapar. Bu hızlı işlemi yavaşlatarak uygulamayı başardığımız ölçüde bilgisayar programlamada başarılı oluruz. Bir de her bilgisayarların dış dünyadan (kendi iç dünyasından dışı) bilgi ve program kabul edebilmesi, programdaki işlem adımlarını uygulayabilmesi ve sonuçları dış dünyaya verebilmesi için bazı özellikleri ve elemanları olması gerekir. Bu elemanların tümü bir bilgisayarın donanım sistemini oluşturur [2, 5,4,6,7]."

3. Programlamanın Yararları

Programlamanın size getireceği artılardan yararlanmak istiyorsanız öncelikle; kendiniz projeler üretip onları yapmaya çalışabilirsiniz. Yaparken göreceksiniz ki çok şey öğreniyor olacaksınız. Bu yapacağınız projeler piyasada aranan veya oralarda kullanılabilecek tarzda bir çalışma ise size gelecekte iş hayatınızda da katkı sağlayacaktır. Basit manada düşünmeyin; geniş ufukla bakın. C++ ile ne yapabilirim bu bildiğim **cout, cin, int** falanla demeyin. Mesela bir muhasebe programı tamam çok geniş olabilir ama sadece dosyalarla işlem yaptırarak basit bir muhasebe (girdi + çıktı + döküm mesela) programı yazabilirsiniz. Bunu yaparken kullanacak kişinin ihtiyaçlarını düşünüp ona göre geliştirmeye kendinizi zorlarsanız, daha fazla verim alırsınız. Hatta tanıdığınız varsa neler yapıyor ne eksiklikleri var tarzı kullandığı program ve muhasebecinin yaptıkları, yapmak istedikleri hakkında bilgi alarak geri dönüşümlü çalışırsanız çok şeylerin değiştiğini göreceksiniz. Tabii ki bu bir örnek...Şöyle bir etrafınıza bakınız: ne kadar da çok yapılabilecek şey varmış dersiniz.

4. Programlamaya Bakış

Programla dillerinde kullanabileceğimiz özellikler sınırlıdır. "reserved word" denilen kalıpları sabittir, bunların dışına çıkılmaz. Programlamacıdan programında sadece bunlara istenilen istikamete doğru yön vermesi beklenir[6].

Aşağıdaki programlama örneklerinde, program yazılırken ve tasarlanırken programcının programa kendisinden neler ekleyebileceği veya kısaltabileceği vurgulanacaktır.

Her şeyden önce yazılacak olan program kısa ve öz olmalı. Herkes tarafından kullanılabilir olması düşünülmeli. Kulacının ne gibi girdiler girilebileceği ön sezi ile tahmin edilmeli. Gereksiz tekrarlı döngülerden kaçınılmalı. Ör-

neğin, program iki veya üç döngü ile çalışıyor olsa bile bunu mümkün ise daha az döngü ile nasıl yapılabilir düşüncesinde olunmalı. Program içinde kullanılmayan değişkenler tanımlanmamalı.

Şimdi de aşağıda aynı sonucu veren farklı düşünceler ile yazılmış bilgisayar programlarını inceleyelim [7]:

Örnek 1: Girilen bir sayının faktöriyelini hesaplayan program verelim.

```
% I. program
function y=fact(n)
% dosya ismi fact.m olmali
% n! hesaplar
if n > 1
      y=n*fact(n-1);
else
     y=1;
 end

% II. program
function y=fact(n)
 % dosya ismi fact.m olmali
% n! hesaplar
fact=1;
for i=1:n
     fact=i*fact;
end
y=fact;
```

Yukarıdaki her iki programda ancak n=170'e kadar çalışacak

```
>> fact(170)
ans =
  7.2574e+306
sonucunu elde ederiz. Fakat
>> fact(171)
ans =
   Inf
```

Eğer bu fonksiyonu çok hassas hesaplamalarda, örneğin $\sum_{k=1}^{200} \frac{sayý}{n!}$ gibi bir ifadede kullandığımızda, 170'den sonraki kesirlerin toplamı sıfır olacağı kesinlikle açıktır. Programı adım adım derlediğimizde bu rahatlıkla görülecektir.

Bir önceki örnek programdaki faktöriyel programı ancak en son 170 sayısının faktöriyelini bulmuştu. 171 sayısı için sınırı aştığı için **inf** (sonsuz) ibaresini vermişti. Şimdi, burada C(173,171) = 173! /((173-171)!.171!)= 14878 ifadesini hesaplamaktan vazgeçeceğimiz anlamına mı geliyor? Cevap olarak: Kesinlikle hayır. Bunun için çok etkili bir program yazarak **sonsuz / sonsuz = NaN** belirsizliğinden kurtulmamız gerekmektedir. Bu engeli aşağıdaki program ile aşabiliriz.

```
n=input('n sayisini giriniz =');
r=input('r sayisini giriniz =');
cnr2=1;
for k=1:r
   cnr2=cnr2*(n-k+1)/k;
end
ycnr=cnr2

EDU» cnr
n sayisini giriniz =173
r sayisini giriniz =171
ycnr =  14878
```

Tanım: Bir sayının kendisi hariç pozitif bölenlerinin toplamı, sayının kendisini veriyor ise bu sayıya mükemmel sayı denir.

Örnek: 6 = 1+2+3

28 = 1+2+4+7+14

```
function  prft= mukemmel(p)
key=input('girdiginiz sayi asal sayi
ise 1 değilse 2 yaziniz =  ');
if  key==1
   prft=  2^(p-1)*(2^p -1);
else
   disp('programi asal sayi girerek
tekrar calistiriniz')
   return
end
```

```
EDU» mukemmel(3)
girdiginiz sayi asal sayi ise 1
değilse 2 yaziniz =   1

ans =   28
» mukemmel(9)
girdiginiz sayi asal sayi ise 1
değilse 2 yaziniz =  2
programi asal sayi girerek tekrar
calistiriniz
```

Örnek 2: Fibonacci dizisini

1 1 2 3 5 8 13 21 34 ....

üreten bir program yazınız.

```
% I. program
function y =fib(n) % fib.m
fib(1)=1;
fib(2)=1;
for i=2:n
      fib(i+1) = fib(i-1) + fib(i);
end
y=fib;

% II. program
function y =fib(n)
if n>1
      y = fib(n-1) + fib(n-2);
else
y=1;
end

>>fib(8)
ans = 1  1  2  3  5  8  13  21  34
```

Örnek 3 : *4 2 4 2 4 2 4 2 4* .... dizisini üreten bir program yazınız.

```
     % I. program
clear all
n=10;
for k=1:n
if  mod(k,2)==0
   cfr(k) =2;
else
   cfr(k) =4;
end   %  if in sonu
end     % k nin sonu;
 ycafer= cfr'

% II. Program
n=10;
for i=1:n-1
         if rem(i,2)==0
         cf=2;
             else
         cf=4;
             end    % end of  if
         t =cf
     end   %  end of  i loop
% III. Program
  for i=1:n-1
    cf=(3 + (-1)^(i+1)) ;
    x  = cf
end
```

Örnek 4 : 3 3 2 3 3 2 3 3 2 .... dizisini üreten bir program yazınız.

```
n=9;
for k=1:n
    if rem(k,3)==0
    cf(k)=2;
else
    cf(k)=3;
end
end
   cf
```

Örnek 5 : C(n,r)= n! /((n-r)! .r!) hesaplayan bir program yazınız.

```
% I. Program
clear all
n=10;
r=3;
cnr=1;
  for j=1:r
      cnr=cnr*(n-j+1)/j;
  end
   %ycnr=cnr
   fprintf('%12.0f\n',cnr)
```

```
% II. Program
n=input('n degerini giriniz = ');
r=input('r degerini giriniz = ');
f1=fakt(n);
f2=fakt(r);
f3=fakt(n-r);
cnr=f1/(f2*f3);
fprintf('C(n,r) = %12d\n',cnr)

%%%%%%%%%%% fonksiyon dosyası

function  v=fakt(k)
v=1;
for i=1:k
    v=v*i;
end
```

$$u(t) = 2 - \exp(t+1) + \int_{-1}^{t} \exp(t-s)u(s)ds \quad \text{(I)}$$

Örnek 6 : Lineer ikinci tip Volterra integral denklemin (I) nümerik çözümünü TEK döngü ile hesaplayan bir program yazınız.

```
% linear  V2  equation
%      u(t) = 2-exp(t+1) +  \int_
{-1}^{t} exp(t-s) u(s)ds     t\geq 0
%       t \in [-1,1], exact soln  is
u(t)=1 ;
%
%
clear
for k =1:3
h=0.1/(2^(k-1))
t(1)=-1;
tmax= 1;
n=(tmax-t(1) ) /h;
u(1) = 1;
tout=t(1);
uout=u(1).';
isum=0;
exa(1) =1;
ero(1)=0 ;
I(1) =0 ;
for i =1:(n+1)
   t(i+1) = t(1) + i*h;
   if  i==1
      cf =0.5;
   else
      cf =1;
   end
   if i==1
      u(i)= 1 ; %u(1);
      else
         u(i)=( 2 - exp(t(i)+ 1)  +
I(i) )/(1- 0.5*h);
              end
               I(i+1) = exp(h)*(I(i)
+  cf*h*u(i) );
         %%%%%%%%   exact soln
            exa(i+1) = 1;
   ero(i+1)= abs(u(i) - exa(i+1) );
      tout=[tout,t(i)];
   uout=[uout,u(i).'];
   %%%
         if rem(i-1,2^k)==0
       fprintf('%5.2f%19.6e\
n',t(i),ero(i+1) )
         end  % end for if statement
 end   % end for i loop
%plot(tout,uout, 'r'),grid
%xlabel('t')
%ylabel('x(t)')
%title('VI2 Eqns')
   end   % end for m loop
```

Örnek 7 : Lineer ikinci tip Volterra integral denklemin (I) nümerik çözümünü ÇİFT döngülü ve fonksiyon dosyası ile hesaplayan bir program yazınız.

```
% linear  V2  equation   with
rectangle rule
%      u(t) = t + 0.2 \int_{0}^{t}
t*s u(s)ds     t\geq 0
%       t \in [0,2]
%
%
clear
h=0.05;
t(1)= 0;
tmax=  2;
```

```
n=(tmax-t(1) ) /h;
u(1) = 0;
tout=t(1);
uout=u(1).';
isum=0;
exa(1) = 0;
ero(1) =0 ;
I(1)   =  0 ;
for i =1:(n+1)
   t(i+1) = t(1) + i*h;
   for j=1:i
      t(j+1) = t(1) + j*h ;
   if i==1
      u(i)=u(1);
      else
         u(i)=( t(i) + I(i) ) ;
      end
         I(j+1) = I(j) +
h*u(j)*gts(t(i),t(j) ) ;
         end
         %%%%%%%%   exact soln is:
            exa(i+1) =
t(i)*exp(t(i)^3 / 15) ;
   ero(i+1)= abs(u(i) - exa(i+1) );
      tout=[tout,t(i)];
   uout=[uout,u(i).'];
   %%%
         if rem(i-1,2)==0
       fprintf('%5.2f%19.6d\
n',t(i),ero(i+1) )
         end  % end for if statement
 end   % end for i loop
plot(tout,uout, 'r'),grid
xlabel('t')
ylabel('x(t)')
title('VI2 Eqns')

%%%%%%%%%%%%%%%%%%%%%%%%
function  yx = gts(x,s)
y =  t*s/5;
```

$$u'(t) = f(t) + \lambda \int_0^t u(s)ds \qquad \text{(II)}$$

$$u(0) = u_0$$

Örnek 9 : Lineer Volterra integro-diferansiyel denklemin (II) nümerik çözümünü Tek döngülü ve fonksiyon dosyası ile hesaplayan bir program yazınız.

```
%%    Theta    Method for  integro-
differential  equation
%    y'(x) =  hom  -  \int_{0}^{t}
y(s)  ds ;   t>0,
%     y(0) = y_0  = 1 - hom ,
y(t) = hom*sin(t) + (1-hom)*cos(t)
%
%%
clear
hom =1 ;
th= 0.5 ;
  for  m = 1:3
 h=0.2/(2^m)  ;
 xmax =  1   ;
 n= xmax/h  ;
x(1)=  0    ;
y(1)=  1-hom ;
tout=x(1)   ;
 yout=y(1).';
true(1)= y(1) ;
ero(1)=0    ;
I(1) =0;
disp('  t          numerical
exact            error     ')
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
for i=1:(n+1)
   x(i+1) = x(1) + i*h  ;
   c= 1 + 0.5*th*h^2  ;
 y(i+1)=( y(i)+h*( (1-th)*(hom -
I(i))+ th*(hom -0.5*h*y(i) - I(i)
)))/ c;
         I(i+1) = I(i) + 0.5*h*y(i)
+ 0.5*h*y(i+1) ;
 true(i+1)= hom * sin(x(i)) +
(1-hom)*cos(x(i))   ;
 ero(i+1) = abs(y(i)-true(i+1));
 tout=[tout;x(i)];
 yout=[yout,y(i).'] ;
 if rem(i-1,2^(m-1))==0
 %fprintf('%5.2f%17.7f%17.7f%17.4e\
n' ,x(i),y(i),true(i+1),ero(i+1) );
```

```
 fprintf('%5.2f%17.7f%17.7f%17.2e\n'
,x(i),y(i),true(i+1),ero(i+1) );
        end
     end
         end  %  end for  m loop
plot(tout,yout,'r'),grid
xlabel('t=time')
ylabel('u(t)= population size')
```

Örnek 10 : Lineer Volterra integro-diferansiyel denklemin (II) nümerik çözümünü ÇİFT döngülü ve fonksiyon dosyası ile hesaplayan bir program yazınız.

```
%%    Theta    Method for  integro-
differential  equation
%    y'(x) =  hom  -  \int_{0}^{t}
y(s)  ds ;   t>0,
%     y(0) = y_0  = 1 - hom ,
y(t) = hom*sin(t) + (1-hom)*cos(t)
%
clear
hom =1 ;
th= 0.5 ;
  for  m = 1:3
 h=0.2/(2^m)  ;
 xmax =  1   ;
 n= xmax/h  ;
x(1)=  0    ;
y(1)=  1-hom ;
tout=x(1)   ;
 yout=y(1).';
true(1)= y(1) ;
ero(1)=0    ;
I(1) =0;
disp('  t          numerical
exact            error     ')
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~')
for i=1:(n+1)
   x(i+1) = x(1) + i*h  ;
   for j=1:i
      x(j+1) = x(1) + j*h ;
   c= 1 + 0.5*th*h^2  ;
 y(i+1)=( y(i)+h*( (1-th)*(hom -
I(i))+ th*(hom -0.5*h*y(i) - I(i)
)))/ c;
         %I(i+1) = I(i) + 0.5*h*y(i)
+ 0.5*h*y(i+1) ;
   I(j+1)= I(j)+h*( gts(x(i+1),x(j))
*y(j)+gts(x(i+1),x(j+1))*y(j+1) )/2;
   %end
 true(i+1)= hom * sin(x(i)) +
(1-hom)*cos(x(i))   ;
 ero(i+1) = abs(y(i)-true(i+1));
 tout=[tout;x(i)];
 yout=[yout,y(i).'] ;
   end
 if rem(i-1,2^(m-1))==0
 %
 fprintf('%5.2f%17.7f%17.7f%17.2e\n'
,x(i),y(i),true(i+1),ero(i+1) );
        end
     end
         end  %  end for  m loop
plot(tout,yout,'r'),grid
xlabel('t=time')
ylabel('u(t)= population size')
%%%%%%%%%%%%%%%%
function  yx = gts(x,s)
yx=1;
```

5. Sonuçlar

Öğrenciler, Internet üzerindeki formlarda, tartışma sitelerinde, ya sınav için ya da ödev için yardım istemektedirler. Bunlar kesinlikle öğrenmek için değil, günü kurtarma ve verilen ödevden veya projeden o an için sadece geçerli not almak içindir. Bir ay sonraki ara sınav, final sınavında veya bir hafta sorulsa çoktan unutmuşlardır bile.

Ayrıca **Algoritma** ve **açık kod** kavramlarını da iyi oturtmak lazım!

Algoritma her zaman paylaşılmalıdır ve bu yeni ufuklara yelken açılmasını sağlamaktadır. Bununla birlikte açık kod, her zaman hazıra konma rahatlığı verdiği için öğrencileri yerinde saydıracak ve daha kötüsü, bu hazırcılık öğrencilerin ilerlemesine değil aksine gerilemesine neden olacaktır [1].

Eğitimcilik dönemimiz boyunca şunu gözlemledik ki; tüm eğitim süresi boyunca aynı öğrenci her ders hala aynı yardımı istiyorsa, yardım ediliyorsa veya ediyorsak demek ki bir yerlerde bir sorun vardır demektir. Normal koşullarda bir üst sınıfa gelmiş bir öğrencinin alt sınıftaki başka öğrencilere yardım edebilmesi en doğal beklentidir. Bunun belli başlı sebebi öğrencilere açık kod vermekten kaynaklanmaktadır. Eğer algoritma verilmiş olsaydı bu sorunların ortaya çıkacağını sanmıyoruz [7].

Kaynaklar

[1] Vatansever, F., 'Algoritma Geliştirme ve Programlamaya Giriş', Seçkin yayıncılık, Ankara, 2002.

[2] Arabacıoğlu, T., Bülbül, H. İ., Filiz, A., 'Başlık, Bilgisayar Programlama Öğretiminde Yeni Bir Yaklaşım', Akademik Bilişim' 07, 31 Ocak–2 Şubat 2007, Kütahya, http://ab.org.tr/ab07/bildiri/99.doc

[3] Çölkesen, R., 'Programlama Sanatı Algoritmalar C Dili Uyarlaması Cilt 1', Papatya yayınları, 2004.

[4] Larry Nyhoff , Sanford Leestma, 'Fortran 77 and Numerical Methods for Engineers and Scientists ', Prentice Hall, New Jersey, 1995.

[5] William J. Palm, 'MATLAB for Engineering Applications', McGraw-Hill, London, 1999.

[6] Brian Hahn, 'Essential MATLAB for Scientists and Engineers', John Wiley & Sons Inc., New York, 1997.

[7] Filiz, A., 'Programlamaya Giriş', Ders Notu, Aydın, 2001.