WYDRUK PROGRAMÓW

```
function [ y ] = f( x )
    y = 1.15 * sin(x) + 2*log(x+2) - 5.5;
end
```

```
% funkcja wyznaczajaca pierwiastek rownania metoda bisekcji
% PARAMETRY:
% f - zadana funkcja
% xl - lewa granica przedzialu
% xr - prawa granica przedzialu
% iterMax - zadana liczba iteracji
function [ x, y, t ] = getBisMethSol( f, xl, xr, iterMax )
    tic;
    % wyznaczam wartosci poczatkowe zmiennych a, b, fa, fb
    a = xl;
    b = xr;
    fa = f(a);
    fb = f(b);
    % alokuje pamiec na wektory rozwiazan
    x = zeros(iterMax, 1);
    y = zeros(iterMax, 1);
    t = zeros(iterMax, 1);
    if fa * fb > 0
        fprintf('Nie ma gwarancji ze w podanym przedziale jest miejsce
zerowe');
        return
    end
    % glowna petla algorytmu, w kazdej iteracji polozenie miejsca
    % zerowego jest poprawiane
    for i = 1 : iterMax
        xm = a + 0.5*(b-a);
        fm = f(xm);
        % uaktualniam zmienne
        if fa * fm < 0
            b = xm;
            fb = fm;
        elseif fm * fb < 0
            a = xm;
            fa = fm;
        end
        % uzupelnienie wektora wynikow
        if abs(fa) < abs(fb)
            x(i,1) = a;
            y(i,1) = fa;
        else
            x(i,1) = b;
            y(i,1) = fb;
        end
        t(i,1) = toc;
    end
end
```

```
% funkcja wyznaczajaca pierwiastek rownania metoda siecznych
% PARAMETRY:
% f - zadana funkcja
% xl - lewa granica przedzialu
% xr - prawa granica przedzialu
% iterMax - zadana liczba iteracji
function [ x, y, t ] = getSecantMethSol( f, xl, xr, iterMax)
    tic;
    % wyznaczam wartosci poczatkowe zmiennych a, b, fa, fb
    a = xl;
    b = xr;
    fa = f(a);
    fb = f(b);
    % alokuje pamiec na wektory rozwiazan
    x = zeros(iterMax, 1);
    y = zeros(iterMax, 1);
    t = zeros(iterMax, 1);
    % glowna petla algorytmu, w kazdej iteracji polozenie miejsca
    % zerowego jest poprawiane
    if fa * fb > 0
        fprintf('Nie ma gwarancji ze w podanym przedziale jest miejsce
zerowe');
        return
    end
    for i = 1 : iterMax
        c = b - ((fb * (b-a))/(fb - fa));
        fc = f(c);
        % uaktualniam zmienne
        a = b;
        fa = fb;
        b = c;
        fb = fc;
        % uzupelnienie wektora wynikow
        if abs(fa) < abs(fb)
            x(i,1) = a;
            y(i,1) = fa;
        else
            x(i,1) = b;
            y(i,1) = fb;
        end
        t(i,1) = toc;
    end
end
```

```
% glowna funkcja obliczajaca pierwiastki
function [ ] = getTask1Sol( )
    iter = 15;
    % ** DLA PRZEDZIALOW MONOTONICZNYCH **
    % przedzial dla pierwszego pierwiastka
    xl1 = 4.46;
    xr1 = 8.01;
    % przedzial dla drugiego pierwiastka
    xl2 = 8.04;
    xr2 = 10.7;
    % PIERWSZY PIERWIASTEK
    % metoda bisekcji
    [b1x, b1y, b1t] = getBisMethSol(@f,xl1,xr1,iter);
    % metoda siecznych
    [s1x, s1y, s1t] = getSecantMethSol(@f,xl1,xr1,iter);
    % DRUGI PIERWIASTEK
```

```
% metoda bisekcji
[b2x, b2y, b2t] = getBisMethSol(@f,xl2,xr2,iter);
% metoda siecznych
[s2x, s2y, s2t] = getSecantMethSol(@f,xl2,xr2,iter);
% ** DLA MAKSYMALNYCH PRZEDZIALOW W KTORYCH OBA SPOSOBY ZBIEGAJA DO
% ROZWIAZANIA **
% przedzial dla pierwszego pierwiastka
xxl1 = 0;
xxr1 = 8.2;
% przedzial dla drugiego pierwiastka
xxl2 = 7.5;
xxr2 = 12.1;
% PIERWSZY PIERWIASTEK
% metoda bisekcji
[bb1x, bb1y, bb1t] = getBisMethSol(@f,xxl1,xxr1,iter);
% metoda siecznych
[ss1x, ss1y, ss1t] = getSecantMethSol(@f,xxl1,xxr1,iter);
% DRUGI PIERWIASTEK
% metoda bisekcji
[bb2x, bb2y, bb2t] = getBisMethSol(@f,xxl2,xxr2,iter);
% metoda siecznych
[ss2x, ss2y, ss2t] = getSecantMethSol(@f,xxl2,xxr2,iter);

fprintf('PIERWSZY PIERWIASTEK - PRZEDZIAL (4.46;8.01)\n');
fprintf('\n');
fprintf('metoda bisekcji\n');
for i = 1 : iter
    fprintf('Iteracja: %d\n', i);
    fprintf('Miejsce zerowe: %g\n', b1x(i));
    fprintf('Wartosc funkcji: %g\n',b1y(i));
    fprintf('Czas: %g\n',b1t(i));
end
fprintf('\n');
fprintf('metoda siecznych\n');
for i = 1 : iter
    fprintf('Iteracja: %d\n', i);
    fprintf('Miejsce zerowe: %g\n', s1x(i));
    fprintf('Wartosc funkcji: %g\n',s1y(i));
    fprintf('Czas: %g\n',s1t(i));
end
fprintf('\n');
fprintf('DRUGI PIERWIASTEK - PRZEDZIAL (8.04;10.7)\n');
fprintf('\n');
fprintf('metoda bisekcji\n');
for i = 1 : iter
    fprintf('Iteracja: %d\n', i);
    fprintf('Miejsce zerowe: %g\n', b2x(i));
    fprintf('Wartosc funkcji: %g\n',b2y(i));
    fprintf('Czas: %g\n',b2t(i));
end
fprintf('\n');
fprintf('metoda siecznych\n');
for i = 1 : iter
    fprintf('Iteracja: %d\n', i);
    fprintf('Miejsce zerowe: %g\n', s2x(i));
    fprintf('Wartosc funkcji: %g\n',s2y(i));
    fprintf('Czas: %g\n',s2t(i));
end
fprintf('PIERWSZY PIERWIASTEK - PRZEDZIAL (0;8.2)\n');
fprintf('\n');
fprintf('metoda bisekcji\n');
```

```
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %g\n', bb1x(i));
        fprintf('Wartosc funkcji: %g\n',bb1y(i));
        fprintf('Czas: %g\n',bb1t(i));
    end
    fprintf('\n');
    fprintf('metoda siecznych\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %g\n', ss1x(i));
        fprintf('Wartosc funkcji: %g\n',ss1y(i));
        fprintf('Czas: %g\n',ss1t(i));
    end
    fprintf('\n');
    fprintf('DRUGI PIERWIASTEK - PRZEDZIAL (7.5;12.1)\n');
    fprintf('\n');
    fprintf('metoda bisekcji\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %g\n', bb2x(i));
        fprintf('Wartosc funkcji: %g\n',bb2y(i));
        fprintf('Czas: %g\n',bb2t(i));
    end
    fprintf('\n');
    fprintf('metoda siecznych\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %g\n', ss2x(i));
        fprintf('Wartosc funkcji: %g\n',ss2y(i));
        fprintf('Czas: %g\n',ss2t(i));
    end
end
```

```
% funkcja rysujaca wykres wraz z miejscami zerowymi dla zad I
function [] = getGraphSol1()
    x = -0.5 : 0.1 : 12.5;
    y = f(x);
    z1 = 7.379;
    z2 = 8.704;
    pp1 = 4.46;
    fpp1 = f(pp1);
    kp1 = 8.01;
    fkp1 = f(kp1);
    pp2 = 0;
    fpp2 = f(pp2);
    kp2 = 8.2;
    fkp2 = f(kp2);
    pd1 = 8.04;
    fpd1 = f(pd1);
    kd1 = 10.7;
    fkd1 = f(kd1);
    pd2 = 7.5;
    fpd2 = f(pd2);
    kd2 = 12.1;
    fkd2 = f(kd2);
    % rysowanie wykresu

plot(x,y,z1,0,'rs',z2,0,'rs',pp1,fpp1,'bo',kp1,fkp1,'bo',pd1,fpd1,'mo',kd1,
fkd1,'mo',pp2,fpp2,'rd',kp2,fkp2,'rd',pd2,fpd2,'kh',kd2,fkd2,'kh');
end
```

```
function [ y ] = f2( x )
    y = -x^4 + 4*x^3 + 7*x^2 + 3*x + 3;
end
```

```
function [ y ] = df2( x )
    y = - 4*x^3 + 12*x^2 + 14*x + 3;
end
```

```
function [ y ] = ddf2( x )
    y = - 12*x^2 + 24*x + 14;
end
```

```
function [] = getGraphSol2()
    x = -2 : 0.1 : 7;
    y = zeros(1,size(x,2));
    for i = 1 : size(x,2)
        y(1,i) = f2(x(1,i));
    end
    z1 = -1.2445;
    z2 = 5.4142;
    plot(x,y,z1,0,'rs',z2,0,'rs');
end
```

```
function [] = getComplexGraph2()
    rex1 = -1.24445;
    imx1 = 0;
    rex2 = 5.41415;
    imx2 = 0;
    rex3 = -0.08485;
    imx3 = 0.66186;
    rex4 = -0.08485;
    imx4 = -0.66186;
    plot(rex1,imx1,'rs',rex2,imx2,'rs',rex3,imx3,'rs',rex4,imx4,'rs');
    xlabel('real');
    ylabel('imaginary');
end
```

```
% funkcja wyznaczajaca pierwiastek rownania metoda Newtona
% PARAMETRY:
% f - zadana funkcja
% fp - pochodna zadanej funkcji
% xb - pierwsze przyblizenie pierwiastka
% iterMax - zadana liczba iteracji
function [ x, y, t ] = getNewtonMethSol( f, fp, xb, iterMax )
    tic;
    % wyznaczam wartosci poczatkowe zmiennych a, fa, fpa
    a = xb;
    fa = f(a);
    fpa = fp(a);
    % alokuje pamiec na wektory rozwiazan
    x = zeros(iterMax, 1);
    y = zeros(iterMax, 1);
    t = zeros(iterMax, 1);
    % glowna petla algorytmu, w kazdej iteracji polozenie miejsca
    % zerowego jest poprawiane
    for i = 1 : iterMax
```

```
        c = a - fa/fpa;
        % uaktualniem zmienne
        a = c;
        fa = f(a);
        fpa = fp(a);
        % uzupelniam wektor wynikow
        x(i,1) = a;
        y(i,1) = fa;
        t(i,1) = toc;
    end
end
```

```
function [ ] = getTask2aSol( )
    iter = 10;
    % PIERWSZY PIERWIASTEK
    x1 = -20;
    [x1r, y1r, t1r] = getNewtonMethSol(@f2,@df2,x1,iter);
    x2 = -10;
    [x2r, y2r, t2r] = getNewtonMethSol(@f2,@df2,x2,iter);
    x3 = 0;
    [x3r, y3r, t3r] = getNewtonMethSol(@f2,@df2,x3,iter);
    % DRUGI PIERWIASTEK
    x4 = 10;
    [x4r, y4r, t4r] = getNewtonMethSol(@f2,@df2,x4,iter);
    x5 = 20;
    [x5r, y5r, t5r] = getNewtonMethSol(@f2,@df2,x5,iter);
    x6 = 30;
    [x6r, y6r, t6r] = getNewtonMethSol(@f2,@df2,x6,iter);

    fprintf('PIERWSZY PIERWIASTEK\n');

    fprintf('Punkt startowy x0 = -20\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %g\n', x1r(i));
        display(x1r(i));
        fprintf('\n');
        fprintf('Wartosc funkcji: %g\n',y1r(i));
        fprintf('Czas: %g\n',t1r(i));
    end
    fprintf('\n');
    fprintf('Punkt startowy x0 = -10\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %g\n', x2r(i));
        fprintf('Wartosc funkcji: %g\n',y2r(i));
        fprintf('Czas: %g\n',t2r(i));
    end
    fprintf('\n');
    fprintf('Punkt startowy x0 = 0\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %g\n', x3r(i));
        fprintf('Wartosc funkcji: %g\n',y3r(i));
        fprintf('Czas: %g\n',t3r(i));
    end
    fprintf('\n');

    fprintf('DRUGI PIERWIASTEK\n');
```

```
    fprintf('Punkt startowy x0 = 10\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %g\n', x4r(i));
        fprintf('Wartosc funkcji: %g\n',y4r(i));
        fprintf('Czas: %g\n',t4r(i));
    end
    fprintf('\n');
    fprintf('Punkt startowy x0 = 20\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %g\n', x5r(i));
        fprintf('Wartosc funkcji: %g\n',y5r(i));
        fprintf('Czas: %g\n',t5r(i));
    end
    fprintf('\n');
    fprintf('Punkt startowy x0 = 30\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %g\n', x6r(i));
        fprintf('Wartosc funkcji: %g\n',y6r(i));
        fprintf('Czas: %g\n',t6r(i));
    end
    fprintf('\n');
end
```

```
% funkcja wyznaczajaca pierwiastek rownania metoda Mullera MM1
% PARAMETRY:
% f - zadana funkcja
% x1 - pierwszy argument
% x2 - drugi argument
% x3 - trzeci argument, pierwsze przyblizenie miejsca zerowego
% iterMax - zadana liczba iteracji
function [ x, y, t ] = getMM1Sol( f, x1, x2, x3, iterMax )
    tic;
    % wyznaczam wartosci poczatkowe zmiennych
    % c - aktualne przyblizenie miejsca zerowego
    a = x1;
    b = x2;
    c = x3;
    fa = f(a);
    fb = f(b);
    fc = f(c);
    % alokuje pamiec na wektory rozwiazan
    x = zeros(iterMax, 1);
    y = zeros(iterMax, 1);
    t = zeros(iterMax, 1);
    % glowna petla algorytmu, w kazdej iteracji polozenie miejsca
    % zerowego jest poprawiane
    for i = 1 : iterMax
        coefA1 = (a - c) * (a - c);
        coefB1 = (a - c);
        coefA2 = (b - c) * (b - c);
        coefB2 = (b - c);
        res1 = fa - fc;
        res2 = fb - fc;
        % wyznaczam wspolczynniki ukladu rownan
        A = (coefB2*res1 - res2*coefB1)/(coefA1*coefB2 - coefB1*coefA2);
        B = (coefA1*res2 - res1*coefA2)/(coefA1*coefB2 - coefB1*coefA2);
        C = fc;
        % wyznaczam rozwiazania ukladu rownan
```

```
        z1 = -2*C/(B + sqrt(B^2 - 4*A*C));
        z2 = -2*C/(B - sqrt(B^2 - 4*A*C));
        % wybieram pierwiastek zmin
        if abs(B + sqrt(B^2 - 4*A*C)) >= abs (B - sqrt(B^2 - 4*A*C))
            zmin = z1;
        else
            zmin = z2;
        end
        % wyznaczam aktualne przyblizenie miejsca zerowego
        d = c + zmin;
        % uaktualniam zmienne
        % odrzucam punkt polozony najdalej od ostatniego wyznaczonego
        % przyblizenia, czyli d
        if abs(d-a) > abs(d-b) && abs(d-a) > abs(d-c)
            a = c;
            fa = fc;
        elseif abs(d-b) > abs(d-a) && abs(d-b) > abs(d-c)
            b = c;
            fb = fc;
        end
        c = d;
        fc = f(c);
        % uzupelnienie wektora wynikow
        x(i,1) = c;
        y(i,1) = fc;
        t(i,1) = toc;
    end
end
```

```
% funkcja wyznaczajaca pierwiastek rownania metoda Mullera MM2
% PARAMETRY:
% f - zadana funkcja
% df - pierwsza pochodna zadane funkcji
% ddf - druga pochodna zadanej funkcji
% xb - pierwsze przyblizenie pierwiastka
% iterMax - zadana liczba iteracji
function [ x, y, t ] = getMM2Sol( f, df, ddf, xb, iterMax)
    tic;
    % wyznaczam wartosci poczatkowe zmiennych
    a = xb;
    fa = f(a);
    dfa = df(a);
    ddfa = ddf(a);
    % alokuje pamiec na wektory rozwiazan
    x = zeros(iterMax, 1);
    y = zeros(iterMax, 1);
    t = zeros(iterMax, 1);
    % glowna petla algorytmu, w kazdej iteracji polozenie miejsca
    % zerowego jest poprawiane
    for i = 1 : iterMax
        z1 = -2*fa / ( dfa + sqrt(dfa^2 - 2*fa*ddfa));
        z2 = -2*fa / ( dfa - sqrt(dfa^2 - 2*fa*ddfa));
        % wybieram pierwiastek zmin
        if abs(dfa + sqrt(dfa^2 - 2*fa*ddfa)) >= abs(dfa - sqrt(dfa^2 -
2*fa*ddfa))
            zmin = z1;
        else
            zmin = z2;
        end
        % uaktualniam zmienne
        a = a + zmin;
```

```
        fa = f(a);
        dfa = df(a);
        ddfa = ddf(a);
        % uzupelnienie wektora wynikow
        x(i,1) = a;
        y(i,1) = fa;
        t(i,1) = toc;
    end
end
```

```
function [  ] = getTask2bMM1Sol(  )
   iter = 10;
   xl1 = -8;
   xm1 = 7;
   xr1 = 14;
   [x1r,y1r,t1r] = getMM1Sol(@f2,xl1,xm1,xr1,iter);
   xl2 = -5;
   xm2 = 3;
   xr2 = 10;
   [x2r,y2r,t2r] = getMM1Sol(@f2,xl2,xm2,xr2,iter);
   xl3 = -10;
   xm3 = -2;
   xr3 = 3;
   [x3r,y3r,t3r] = getMM1Sol(@f2,xl3,xm3,xr3,iter);

   fprintf('xl = -8, xm = 7, xr = 14\n');
   for i = 1 : iter
       fprintf('Iteracja: %d\n', i);
       fprintf('Miejsce zerowe: %s\n', num2str(x1r(i)));
       fprintf('Wartosc funkcji: %s\n',num2str(y1r(i)));
       fprintf('Czas: %g\n',t1r(i));
   end
   fprintf('\n');
   fprintf('xl = -5, xm = 3, xr = 10\n');
   for i = 1 : iter
       fprintf('Iteracja: %d\n', i);
       fprintf('Miejsce zerowe: %s\n', num2str(x2r(i)));
       fprintf('Wartosc funkcji: %s\n',num2str(y2r(i)));
       fprintf('Czas: %g\n',t2r(i));
   end
   fprintf('\n');
   fprintf('xl = -10, xm = -2, xr = 3\n');
   for i = 1 : iter
       fprintf('Iteracja: %d\n', i);
       fprintf('Miejsce zerowe: %s\n', num2str(x3r(i)));
       fprintf('Wartosc funkcji: %s\n',num2str(y3r(i)));
       fprintf('Czas: %g\n',t3r(i));
   end
   fprintf('\n');
end
```

```
function [] = getTask2bMM2Sol( )
    iter = 10;
    % PIERWSZY PIERWIASTEK
    xb1 = -2;
    [x1r, y1r, t1r] = getMM2Sol(@f2,@df2,@ddf2,xb1,iter);
    xb2 = -1;
    [x2r, y2r, t2r] = getMM2Sol(@f2,@df2,@ddf2,xb2,iter);
    % DRUGI/TRZECI PIERWIASTEK
    xb3 = 0;
    [x3r, y3r, t3r] = getMM2Sol(@f2,@df2,@ddf2,xb3,iter);
```

```
    xb4 = 3;
    [x4r, y4r, t4r] = getMM2Sol(@f2,@df2,@ddf2,xb4,iter);
    % TRZECI/CZWARTY PIERWIASTEK
    xb5 = 4;
    [x5r, y5r, t5r] = getMM2Sol(@f2,@df2,@ddf2,xb5,iter);
    xb6 = 5;
    [x6r, y6r, t6r] = getMM2Sol(@f2,@df2,@ddf2,xb6,iter);

    % wypisywanie wynikow
    fprintf('Punkt startowy x0 = -2\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %s\n', num2str(x1r(i)));
        fprintf('Wartosc funkcji: %s\n',num2str(y1r(i)));
        fprintf('Czas: %g\n',t1r(i));
    end
    fprintf('\n');
    fprintf('Punkt startowy x0 = -1\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %s\n', num2str(x2r(i)));
        fprintf('Wartosc funkcji: %s\n',num2str(y2r(i)));
        fprintf('Czas: %g\n',t2r(i));
    end
    fprintf('\n');
    fprintf('Punkt startowy x0 = 0\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %s\n', num2str(x3r(i)));
        fprintf('Wartosc funkcji: %s\n',num2str(y3r(i)));
        fprintf('Czas: %g\n',t3r(i));
    end
    fprintf('\n');

    fprintf('Punkt startowy x0 = 3\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %s\n', num2str(x4r(i)));
        fprintf('Wartosc funkcji: %s\n',num2str(y4r(i)));
        fprintf('Czas: %g\n',t4r(i));
    end
    fprintf('\n');
    fprintf('Punkt startowy x0 = 4\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %s\n', num2str(x5r(i)));
        fprintf('Wartosc funkcji: %s\n',num2str(y5r(i)));
        fprintf('Czas: %g\n',t5r(i));
    end
    fprintf('\n');
    fprintf('Punkt startowy x0 = 5\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %s\n', num2str(x6r(i)));
        fprintf('Wartosc funkcji: %s\n',num2str(y6r(i)));
        fprintf('Czas: %g\n',t6r(i));
    end
    fprintf('\n');
end
```

```
function [ ] = CompMM2andNewton( )
```

```
    iter = 10;
    x1 = 20;
    [x1r, y1r, t1r] = getNewtonMethSol(@f2,@df2,x1,iter);
    [x2r, y2r, t2r] = getMM2Sol(@f2,@df2,@ddf2,x1,iter);

    fprintf('Metoda Newtona\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %s\n', num2str(x1r(i)));
        fprintf('Wartosc funkcji: %s\n',num2str(y1r(i)));
        fprintf('Czas: %g\n',t1r(i));
    end
    fprintf('\n');
    fprintf('Metoda MM2\n');
    for i = 1 : iter
        fprintf('Iteracja: %d\n', i);
        fprintf('Miejsce zerowe: %s\n', num2str(x2r(i)));
        fprintf('Wartosc funkcji: %s\n',num2str(y2r(i)));
        fprintf('Czas: %g\n',t2r(i));
    end
    fprintf('\n');
end
```