Rozwiązywanie równania nieliniowego metodą Newtona.

1. **Zastosowanie** : Funkcja Newton znajduje przybliżoną wartość pierwiastka równania $f(x) = 0$.
2. **Opis metody** : Pierwiastek równania wyznacza się stosując proces iteracyjny Newtona postaci $x_{(i+1)} = x_{(i)} - \frac{f(x_{(i)})}{f'(x_{(i)})};\ i = 0, 1, \ldots$, w którym wartość x_0 jest dana. Proces kończy się, gdy $\frac{|x_{(i+1)} - x_{(i)}|}{max(|x_{(i+1)}, x_{(i)}|} < \varepsilon,\ x_{(i+1)} \neq 0\ lub\ x_{(i)} \neq 0,$ gdzie ε oznacza zadaną z góry dokładność lub gdy $x_{(i+1)} = x_{(i)} = 0.$
3. **Wywołanie** :
   a. arytmetyka zmiennopozycyjna : $Newton$ (x, f, df, mit, eps, fatx, it, st);
   b. arytmetyka przedziałowa : $NewtonInterval$ (x, f, df, mit, eps, fatx, it, st);
4. **Dane** :
   a. x - początkowe przybliżenie pierwiastka
   b. f - funkcja języka Turbo Pascal, która dla danej wartości x oblicza $f(x)$
   c. df - funkcja języka Turbo Pascal, która dla danej wartości x oblicza $f'(x)$
   d. mit - maksymalna liczba iteracji w procesie
   e. eps - błąd względny wyznaczania pierwiastka
5. **Wyniki** :
   a. $Newton$ (x, f, df, mit, eps, fatx, it, st); - przybliżona wartość pierwiastka
   b. $NewtonInterval$ (x, f, df, mit, eps, fatx, it, st); - przybliżona wartość pierwiastka
   c. $fatx$ - wartość funkcji f dla obliczonej wartości pierwiastka
   d. it - liczba wykonanych operacji
6. **Inne parametry** :
   a. st - zmienna, która po wykonaniu funkcji $Newton$ lub $NewtonInterval$ ma jedną z następujących wartości :
      i. 1, jeżeli $mit < 1$
      ii. 2, gdy podczas obliczeń $f'(x) = 0$ dla pewnej wartości x
      iii. 3, jeżeli w mit krokach iteracyjnych nie osiągnięto podanej dokładności ε
      iv. 0, w przeciwnym przypadku
7. **Typy parametrów** :
   a. Integer : $it,\ mit,\ st$
   b. Extended : $eps,\ fatx,\ x$
   c. fx : $f,\ df$
   d. *Interval: $eps,\ fatx,\ x$ dla arytmetyki przedziałowej
8. **Identyfikator nielokalny** :
   a. fx - identyfikator typu proceduralnego zdefiniowany następująco :

```
type fx = function (x : Extended) : Extended; far;
```

b. *fxinterval* - identyfikator typu proceduralnego zdefiniowany następująco :
type fxinterval = function (x : interval; var st : Integer) : interval; far;
zmienna *st* służy do raportowania o stanie wykonania funkcji np. funkcja iSin(x,st) zwraca wartość stanu, może nie poprawnego wyniku.

9. **Kod źródłowy** :
   a. *NewtonMethod.pas*

```
unit NewtonMethod;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, Menus, ComCtrls, StdCtrls, Grids, Clipbrd,
IntervalArithmetic32and64, Math;

type fx = function (x : Extended) : Extended; far;

function Newton(var x : Extended; f : fx; df : fx;  mit : Integer; eps : Extended; var fatx
: Extended; var it : Integer; var st : Integer) : Extended;

implementation

function Newton(var x : Extended; f : fx; df : fx;  mit : Integer;  eps : Extended; var fatx
: Extended; var it : Integer; var st : Integer) : Extended;
var
  mark : Boolean;
  xit : Extended;  //x po i-tej iteracji
  funkcja : Extended;
  pochodna : Extended;
begin
    mark := true;
   it := 0;
   st := 0;
//   xit := x+1;
   funkcja := 0;
   pochodna := 0;

   if mit < 1 then   //mit jest za mały
    begin
      st := 1;
      fatx := f(x);
      it := 0;
      mark := false;
      Result := x;
    end;
```

```
  while (it < mit) do
   begin
    if it <> 0 then
      begin
        x:=xit;
      end;
    //obliczenie wartości funkcji i pochodnej
    funkcja := f(x);
    pochodna := df(x);

    if pochodna = 0 then     //wartość pochodnej dla x jest równa 0
      begin
        st := 2;
        fatx := f(x);
        mark := false;
        Result := x;
        break;
      end;

    xit := x-(funkcja/pochodna);

    it := it + 1;

    if ((abs(xit-x)/Max(abs(xit),abs(x)))<eps) then
      begin
        break;
      end;
   end;

   if ((it = mit) and ((abs(xit-x)/Max(xit,x)>eps))) then    //nie osiągnietą wymaganej
dokładności po mit iteracjach
    begin
      st := 3;
      fatx := f(xit);
      mark := false;
      Result := xit;
    end;

   if mark then
    begin
      fatx := f(x);
      Result := xit;
    end;
end;
end.
```

b. *NewtonInterval.pas*

```
unit NewtonIntervalMethod;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, Menus, ComCtrls, StdCtrls, Grids, Clipbrd,
IntervalArithmetic32and64, Math;

type fxinterval = function (x : interval; var st : Integer) : interval; far;

function NewtonInterval(var x : interval; f : fxinterval; df : fxinterval;  mit : Integer; eps :
interval; var fatx : interval; var it : Integer; var st : Integer) : interval;

implementation

function NewtonInterval(var x : interval; f : fxinterval; df : fxinterval;  mit : Integer; eps :
interval; var fatx : interval; var it : Integer; var st : Integer) : interval;
var
  mark : Boolean;
  xit : interval;  //x po i-tej iteracji
  funkcja : interval;
  pochodna : interval;
  stanFunkcji : Integer;
  maxInterval : interval;
begin
    mark := true;
   it := 0;
   st := 0;
//   xit := x+1;
   funkcja := 0;
   pochodna := 0;

   if mit < 1 then   //mit jest za mały
    begin
      st := 1;
      fatx := f(x, stanFunkcji);
      it := 0;
      mark := false;
      Result := x;
    end;

   while (it < mit) do
    begin
```

```
 if it <> 0 then
   begin
     x.a:=xit.a;
     x.b:=xit.b;
   end;
 //obliczenie wartości funkcji i pochodnej
  funkcja := f(x,stanFunkcji);
  if stanFunkcji <> 0 then
   begin
     st:= 4;
     showMessage('Błąd funkcji');
     break;
   end;
  pochodna := df(x, stanFunkcji);
  if stanFunkcji <> 0 then
   begin
     st := 5;
     showMessage('Błąd pochodnej');
     break
   end;


  if containtZero(pochodna) then     //wartość przedziału pochodnej dla x zawiera 0
   begin
     st := 2;
     fatx := f(x, stanFunkcji);
     mark := false;
     Result := x;
     break;
   end;

  xit := x-(funkcja/pochodna);

  it := it + 1;

  if iabs(xit)>iabs(x) then maxInterval := iabs(xit)
  else maxInterval := iabs(x);

  if containtZero(maxInterval) then
   begin
     st := 2;
     fatx := f(x, stanFunkcji);
     mark := false;
     showMessage('Przy obliczaniu błędu mogło dojść do dzielenia przez zero');
     Result := x;
     break;
```

```
      end;

    if ((iabs(xit-x)/maxInterval)<eps) then
      begin
        break;
      end;
  end;

  if ((it = mit) and ((iabs(xit-x)/maxInterval)>eps)) then    //nie osiągnietą wymaganej
dokładności po mit iteracjach
    begin
      st := 3;
      fatx := f(xit, stanFunkcji);
      mark := false;
      Result := xit;
    end;

  if mark then
    begin
      fatx := f(x, stanFunkcji);
      Result := xit;
    end;
end;
end.
```

10. **Przykłady** :

Dane w postaci liczb rzeczywistych :

a) **Równanie:** $x^2 - 2 = 0$

Definicje funkcji f i df:

```
function f (x : Extended) : Extended; far;
begin
  f :=x*x-2;
end;
function df (x : Extended) : Extended; far;
begin
  df := 2*x;
end;
```

Dane:

$x = 1,\ mit = 100,\ eps = 10^{-16}$

Wyniki :

$Newton$ (x, f, df, mit, eps, fatx, it, st) = 1.4142135623731

$fatx = -1.0842021724855E-19$

$it = 6$

$st = 0$

b) **Równanie - jak w przykładzie a)**

Dane:

$x = 0,\ mit = 10,\ eps = 10^{-16}$

Wyniki :

$st = 2$

c) **Równanie - jak w przykładzie a)**

Dane:

$x = 4.5,\ mit = 5,\ eps = 10^{-16}$

Wyniki:

$Newton$ (x, f, df, mit, eps, fatx, it, st) = 1.41421356494674

$fatx = 7.277936684649828E - 9$

$it = 5$

$st = 3$

d) **Równanie :** $sin^2(x) + 0.5sin(x) - 0.5 = 0$

Definicje funkcji f i df:

```
function f (x : Extended) : Extended; far;
  var s : Extended;
  begin
    s:=Sin(x);
    f:=s*(s+0.5)-0.5;
end;
function df (x : Extended) : Extended; far;
  begin
    df:=Sin(2*x) + 0.5*Cos(x);
  end;
```

Dane :

$x = 0.6,\ mit = 20,\ eps = 10^{-16}$

Wyniki :

$Newton$ (x, f, df, mit, eps, fatx, it, st) = 0.523598775598299

$fatx = 0$

$it = 5$

$st = 0$

Dane w postaci przedziałów:

a) **Równanie:** $x^2 - 2 = 0$

Definicje funkcji f i df:

```
function f (x : interval; var st :Integer) : interval; far;
begin
  f :=x*x-2;
  st := 0;
end;
function df (x : interval; var st : Integer) : interval; far;
```

```
begin
  df := 2*x;
  st := 0;
end;
```

Dane:

$x = [1.0E+0;\ 1.0E+0]$

$mit = 100$

$eps = [1.0000000000000000E-16;\ 1.0000000000000001E-16]$

Wyniki :

$NewtonInterval$ (x, f, df, mit, eps, fatx, it, st) =

$[1.4142135623730950E+0;\ 1.4142135623730951E+0]$

$fatx = [-3.2526065174565134E-18;\ 3.4694469519536142E-18]$

$it = 6$

$st = 0$

b) **Równanie - jak w przykładzie a)**

Dane:

$x = [0.0E+0;\ 0.0E+0]$

$mit = 10$

$eps = [1.0000000000000000E-16;\ 1.0000000000000001E-16]$

Wyniki :

$st = 2$

c) **Równanie - jak w przykładzie a)**

Dane:

$x = [4.5E+0;\ 4.5E+0]$

$mit = 5$

$eps = [1.0000000000000000E-16;\ 1.0000000000000001E-16]$

Wyniki:

$NewtonInterval$ (x, f, df, mit, eps, fatx, it, st) =

$[1.4142135649467398E+0;\ 1.4142135649467399E+0]$

$fatx = [7.2793668367404640E-9;\ 7.2793668560392628E-9]$

$it = 5$

$st = 3$

d) **Równanie :** $sin^2(x) + 0.5sin(x) - 0.5 = 0$

Definicje funkcji f i df:

```
function f (x : interval; var st : Integer) : interval; far;
  var
    s : interval;
  begin
    s:=iSin(x,st);
```

```
    f:=s*(s+0.5)-0.5;
end;
function df (x : interval; var st : Integer) : interval; far;
var
  st2 : Integer;
  begin
    df:=iSin(2*x, st) + 0.5*iCos(x, st2);
    if st2 <> 0 then st := st2;
  end;
```

Dane :

$x = [5.9999999999999999E-1;\ 6.0000000000000001E-1]$

$mit = 20$

$eps = [1.0000000000000000E-16;\ 1.0000000000000001E-16]$

Wyniki :

$NewtonInterval$ (x, f, df, mit, eps, fatx, it, st) =

$[5.2349877559829885E-1;\ 5.2359877559829890E-1]$

$fatx = [-1.3227266504323155E-17;\ 1.3227266504323155E-17]$

$it = 5$

$st = 0$