Отчет к лабораторной работе №3

Круглов В.А.

Main.pas

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Math;

type
  TMainForm = class(TForm)
    GridPanel: TGridPanel;
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    Panel5: TPanel;
    mSource: TMemo;
    Panel3: TPanel;
    Label1: TLabel;
    btnAction: TButton;
    Panel4: TPanel;
    eResult: TEdit;
    Panel2: TPanel;
    GroupBox2: TGroupBox;
    Panel6: TPanel;
    mPostfix: TMemo;
    Panel7: TPanel;
    GroupBox3: TGroupBox;
    Panel8: TPanel;
    mConverted: TMemo;
    procedure btnActionClick(Sender: TObject);
  private
    NumberStack: array [0 .. $FFFF] of real;
    CommandStack: array [0 .. $FFFF] of char;
    NumberCounter, CommandCounter: word;
    procedure PushNumber(aNumber: real); inline;
    procedure PushCommand(aCommand: char); inline;
    procedure Reset;
    function PopNumber: real; inline;
    function PopCommand: char; inline;
    function PrepareClause(aClause: string): string;
    function DoCommand(aCommand: char; N1, N2: real): real;

  public
    { Public declarations }
  end;

var
  MainForm: TMainForm;
  ErrorFlag: byte;

implementation

uses ClassesUnit;
{$R *.dfm}

procedure TMainForm.btnActionClick(Sender: TObject);
const
  prs = '+-*/^(';
  pri: array [1 .. 6] of byte = (1, 1, 2, 2, 3, 0);
var
  cflag: boolean;
  c: char;
```

```
  s, sNumber, r: String;
  i: longword;
begin
  s := PrepareClause(mSource.Text);
  mConverted.Text := s;
  sNumber := '';
  ErrorFlag := $00;
  Reset;
  cflag := false;
  for i := 1 to Length(s) do
    case s[i] of
      '0' .. '9', '.':
        begin
          sNumber := sNumber + s[i];
          cflag := false;
        end;
      '(':
        PushCommand(s[i]);
      ')':
        begin
          if cflag then
          begin
            mPostfix.Text := 'Error: unexpected '')''';
            eResult.Text := 'Error: convert error';
            Exit;
          end;
          if sNumber <> '' then
          begin
            PushNumber(StrToFloat(sNumber));
            r := r + sNumber + ' ';
            sNumber := '';
          end;
          c := PopCommand;
          while (c <> '(') and (c <> #$00) do
          begin
            r := r + c + ' ';
            PushNumber(DoCommand(c, PopNumber, PopNumber));
            c := PopCommand;
          end;
          if c = #$00 then
          begin
            mPostfix.Text := 'Error: unexpected '')''';
            eResult.Text := 'Error: convert error';
            Exit;
          end;
        end;
      '+', '-', '*', '/', '^':
        begin
          if cflag then
          begin
            mPostfix.Text := 'Error: unexpected ''' + s[i] + ''''';
            eResult.Text := 'Error: convert error';
            Exit;
          end;
          if sNumber <> '' then
          begin
            PushNumber(StrToFloat(sNumber));
            r := r + sNumber + ' ';
            sNumber := '';
          end;
          c := PopCommand;
          while c <> #$00 do
            if pri[Pos(c, prs)] >= pri[Pos(s[i], prs)] then
            begin
              r := r + c + ' ';
              PushNumber(DoCommand(c, PopNumber, PopNumber));
              c := PopCommand;
            end
            else
              break;
```

```
          PushCommand(c);
          PushCommand(s[i]);
          cflag := true;
        end;
    end;
  if cflag then
  begin
    mPostfix.Text := 'Error: expected number, but not found';
    eResult.Text := 'Error: convert error';
    Exit;
  end;
  if sNumber <> '' then
  begin
    PushNumber(StrToFloat(sNumber));
    r := r + sNumber + ' ';
  end;
  c := PopCommand;
  while c <> #$00 do
    if c = '(' then
    begin
      mPostfix.Text := 'Error: expected '')'', but not found';
      eResult.Text := 'Error: convert error';
      Exit;
    end
    else
    begin
      r := r + c + ' ';
      PushNumber(DoCommand(c, PopNumber, PopNumber));
      c := PopCommand;
    end;
  mPostfix.Text := r;
  case ErrorFlag of
    $00:
      eResult.Text := CurrToStr(PopNumber);
    $01:
      eResult.Text := 'Error: division by ZERO';
    $02:
      eResult.Text := 'Error: negative exponent with non-whole power';
  end;
end;

function TMainForm.DoCommand(aCommand: char; N1, N2: real): real;
begin
  case aCommand of
    '+':
      Result := N2 + N1;
    '-':
      Result := N2 - N1;
    '*':
      Result := N2 * N1;
    '/':
      begin
        if N1 <> 0 then
          Result := N2 / N1
        else
        begin
          if ErrorFlag = $00 then
            ErrorFlag := $01;
          Result := 0;
        end;
      end;
    '^':
      begin
        if (N2 < 0) and (Round(N1) <> N1) then
        begin
          if ErrorFlag = $00 then
            ErrorFlag := $02;
          Result := 0;
        end
        else
```

```
          Result := Power(N2, N1);
      end;
  end;
end;

function TMainForm.PopCommand: char;
begin
  if CommandCounter = 0 then
  begin
    Result := #$00;
    Exit;
  end;
  Dec(CommandCounter);
  Result := CommandStack[CommandCounter];
end;

function TMainForm.PopNumber: real;
begin
  Dec(NumberCounter);
  Result := NumberStack[NumberCounter];
end;

function TMainForm.PrepareClause(aClause: string): string;
var
  i: longword;
  SubCounter: byte;
   SubFlag: ShortInt;
  ExpFlag: boolean;
begin
  Result := '';
  SubCounter := 0;
  ExpFlag := false;
  SubFlag := 0;
  for i := 1 to Length(aClause) do
  begin
    if (aClause[i] in ['-', '+']) and (SubFlag = 0) and
      ((i = 1) or (aClause[i - 1] in ['*', '/', '^', '(', '+', '-'])) then
    begin
      if aClause[i] = '-' then
        Inc(SubCounter);
    end
    else if (aClause[i] = '*') and (i < Length(aClause)) and
      (aClause[i + 1] = '*') then
    begin
      ExpFlag := true;
    end
    else
      case aClause[i] of
        '0' .. '9', '.':
          begin
            if SubFlag > 0 then
              Result := Result + '+'
            else if SubFlag < 0 then
              Result := Result + '-';
            if SubFlag <> 0 then
              SubFlag := 0;
            if SubCounter mod 2 > 0 then
              Result := Result + '(0-' + aClause[i]
            else
              Result := Result + aClause[i];
          end;
        '+', '-', '*', '/', '^', '(', ')':
          begin
            if (SubCounter > 0) and (SubCounter mod 2 > 0) then
            begin
              Result := Result + ')';
              SubCounter := 0;
            end;
            if (aClause[i] = '*') and ExpFlag then
            begin
```

```
              aClause[i] := '^';
              ExpFlag := false;
            end;
            if aClause[i] = '+' then
            begin
              if SubFlag = 0 then
                SubFlag := 1;
            end
            else if aClause[i] = '-' then
              if SubFlag = 0 then
                SubFlag := -1
              else
                SubFlag := -SubFlag
              else
              begin
                if SubFlag > 0 then
                  Result := Result + '+'
                else if SubFlag < 0 then
                  Result := Result + '-';
                Result := Result + aClause[i];
                if SubFlag <> 0 then
                  Exit;
              end;
          end;
        ',':
          begin
            if SubCounter mod 2 > 0 then
              Result := Result + '(0-' + '.'
            else
              Result := Result + '.';
          end;
      end;
  end;
  if (SubCounter > 0) and (SubCounter mod 2 > 0) then
  begin
    Result := Result + ')';
    Dec(SubCounter);
  end;
end;

procedure TMainForm.PushCommand(aCommand: char);
begin
  if aCommand = #$00 then
    Exit;
  CommandStack[CommandCounter] := aCommand;
  Inc(CommandCounter);
end;

procedure TMainForm.PushNumber(aNumber: real);
begin
  NumberStack[NumberCounter] := aNumber;
  Inc(NumberCounter);
end;

procedure TMainForm.Reset;
begin
  CommandCounter := 0;
  NumberCounter := 0;
end;

end.
```