```
unit LCRPreCompile;

interface

uses System.SysUtils, Classes, DateUtils;

type
  TLCRPreCompile = class
  private
    fSourcePath, fWorkBook, fWorkBookBase, tmpString, fBook : string;
    fAllVars : TStringList;
    MaxR : integer;
    procedure ReadSteps;
    procedure ExtractNames(s : string);
    procedure AddVars;
    procedure AddLookup;
    procedure LoadVars;
    procedure DumpVars;
  public
    constructor create(aSourcePth : string; aCostWorkbook, aCostWorkbookBase  :
string);
    destructor Destroy; override;

    procedure Go;
  end;

implementation

uses VCL.FlexCel.Core, FlexCel.XlsAdapter;

const NL = #13#10;

{ TLCRPreCompile }

procedure TLCRPreCompile.AddLookup;
var i : integer;
    tmps : string;
const indent = '  ';
begin
  tmps:=indent;
  for i:=0 to fAllVars.Count -1 do begin
    tmps:=tmps+'if s = '+ QuotedStr(fAllVars[i])+' then
_Variables.p_'+fAllVars[i]+':=pd';
    if i<fAllVars.Count -1 then
      tmps:=tmps+' else '+NL+indent
    else
      tmps:=tmps+' ;'+NL;
  end;
  tmpString := StringReplace(tmpString,'(*_SetVarPointer'+fBook+'*)',tmps,
[rfIgnoreCase,rfReplaceAll]);
end;

procedure TLCRPreCompile.AddVars;
var i : integer;
    tmps : string;
const indent = '    ';
begin
  tmps:=indent;
  for i:=0 to fAllVars.Count -1 do begin
    tmps:=tmps+fAllVars[i];
    if i<fAllVars.Count -1 then tmps:=tmps+',';
    if i mod 6 = 1 then tmps:=tmps+NL+indent;
  end;
  tmps:=tmps + ' : double;'+NL+NL+indent;

  for i:=0 to fAllVars.Count -1 do begin
    tmps:=tmps+'P_'+fAllVars[i];
    if i<fAllVars.Count -1 then tmps:=tmps+',';
    if i mod 6 = 1 then tmps:=tmps+NL+indent;
  end;
  tmps:=tmps + ' : pdouble;';
  tmpString := StringReplace(tmpString,'(*VARIABLES'+fBook+'*)',tmps,
[rfIgnoreCase,rfReplaceAll]);
end;

procedure TLCRPreCompile.LoadVars;
var i : integer;
    tmps : string;
const indent = '  ';
begin
  tmps:=indent;
  for i:=0 to fAllVars.Count -1 do begin
    tmps:=tmps+'if Assigned(_Variables.P_'+fAllVars[i]+') then
_Variables.'+fAllVars[i] + ':= _Variables.P_'+fAllVars[i]+'^;'+NL+indent;
  end;
  tmpString := StringReplace(tmpString,'(*LoadVars'+fBook+'*)',tmps,
[rfIgnoreCase,rfReplaceAll]);
end;

constructor TLCRPreCompile.create(aSourcePth, aCostWorkbook, aCostWorkbookBase:
string);
begin
  fSourcePath:=aSourcePth;
  fWorkBook:=aCostWorkbook;
  fWorkBookBase:=aCostWorkbookBase;
end;
```

```
destructor TLCRPreCompile.Destroy;
begin
  inherited;
end;

procedure TLCRPreCompile.DumpVars;
var i : integer;
    tmps : string;
const indent = '  ';
begin
  tmps:=indent;
  for i:=0 to fAllVars.Count -1 do begin
    tmps:=tmps+'s := s + ' +QuotedStr(fAllVars[i] + ': ') +' +
_Variables.'+fAllVars[i]+'.ToString + #13#10;'+NL+indent;
  end;
  tmpString := StringReplace(tmpString,'(*DumpVars'+fBook+'*)',tmps,
[rfIgnoreCase,rfReplaceAll]);
end;

procedure TLCRPreCompile.ExtractNames(s: string);
var i,j : integer;
    v : string;
const
    OKC = ['a'..'z','A'..'Z','0'..'9','_'];
    OKS = ['a'..'z','A'..'Z','_'];
begin
  i:=1;
  while i<=length(s) do begin
    v:='';
    while (not (s[i] in OKC)) and (I<length(s)) do inc(i);
    repeat
      if s[i] in OKC then
        v:=v+s[i];
      inc(i);
      if i > length(s) then break;
    until (not (s[i] in OKC)) and (I<=length(s));
    if length(v)>0 then
      if v[1] in OKS then
        fAllVars.Add(lowercase(V));
  end;
end;

procedure TLCRPreCompile.Go;
var tSL : TStringList;
    v : integer;
begin
  tSL := TSTringList.Create;
  tSL.LoadFromFile(fSourcePath+'LCRCompiledCostTemplate.pas');
  tSL.Strings[0]:='unit LCRCompiledCost;' ;
  tmpString:=tSL.Text;
  tmpString := StringReplace(tmpString,'(*WORKBOOKOPTION*)',fWorkBook,
[rfIgnoreCase,rfReplaceAll]);
  tmpString := StringReplace(tmpString,'(*WORKBOOKBASELINE*)',fWorkBookBase,
[rfIgnoreCase,rfReplaceAll]);
  tmpString := StringReplace(tmpString,'(*DATE*)',DateTimeToStr(Now()),
[rfIgnoreCase,rfReplaceAll]);

  MaxR := 0;
  fBook := 'BASELINE';
  fAllVars := TStringList.Create;
  fAllVars.Sorted := true;
  fAllVars.Duplicates := dupIgnore;
  ReadSteps();
  AddVars();
  AddLookup();
  LoadVars();
  DumpVars();
  tmpString :=
StringReplace(tmpString,'(*NUMVARSBASELINE*)',fAllVars.Count.ToString,
[rfIgnoreCase,rfReplaceAll]);
  fAllVars.Free;

  fBook := 'OPTION';
  fAllVars := TStringList.Create;
  fAllVars.Sorted := true;
  fAllVars.Duplicates := dupIgnore;
  ReadSteps();
  AddVars();
  AddLookup();
  LoadVars();
  DumpVars();
  tmpString := StringReplace(tmpString,'(*NUMVARSOPTION*)',fAllVars.Count.ToString,
[rfIgnoreCase,rfReplaceAll]);
  fAllVars.Free;

  tmpString := StringReplace(tmpString,'9997', MaxR.ToString,
[rfIgnoreCase,rfReplaceAll]);

  tSL.Text := tmpString;
  tSL.SaveToFile(fSourcePath+'LCRCompiledCost.pas');
  tSL.Free;
end;

procedure TLCRPreCompile.ReadSteps;
var
```

```
  Xls: TExcelFile;
  r, ci: integer;
  evString,t,evStringSt,SetState, tmps : string;
  usecc,astate : boolean;
const indent = '    ';
begin
  if fBook = 'BASELINE' then
    Xls := TXlsFile.Create(fWorkBookBase, False)
  else
    Xls := TXlsFile.Create(fWorkBook, False);
  Xls.ActiveSheetByName := 'Steps';
  {
    CWS_Costing_Steps_logic.xlsx
    A 1 Cost Number
    B 2 Cost Name
    C 3 Cost Description
    D 4 Probability cost applies to PWS or state (blank=1)
    E 5 Total Cost per Event (expression)
    F 6 Hours (Reporting)
    G 7 Labor (Reporting)
    H 8 O&M (Reporting)
    I 9 Domain
  }
  ci:=0;
  evString := indent;
  evStringSt := indent;
  SetState := indent;
  for r := 2 to Xls.RowCount do begin
    if not ((Xls.GetStringFromCell(r, 1) <> '') and
       (Xls.GetStringFromCell(r, 2) <> '')) then continue;

     t := Xls.GetStringFromCell(r, 2);
     if t[1]='#' then continue;
     if Xls.GetStringFromCell(r, 9) = 'State' then begin
       astate:=true;
       SetState := SetState + '_ImAState['+ci.ToString+'] := true;' + NL + '  ';
     end else begin
       astate:=false;
       SetState := SetState + '_ImAState['+ci.ToString+'] := false;' + NL + '  ';
     end;

     if aState then
       tmps := evstringSt
     else
       tmps := evstring;

     tmps := tmps + NL + indent;
     tmps := tmps + '//' + t +'  row: ' +r.ToString + NL + indent;
     usecc:=false;

     t:=trim(Xls.GetStringFromCell(r, 4));
     ExtractNames(t);
     if t<>'' then begin
       tmps := tmps + '_CalcCost['+ci.ToString+'] := '+ t + ';'+NL + indent +
              'inc(TotEval);'+NL + indent;
       usecc:=true;
       tmps := tmps + 'if _CalcCost['+ci.ToString+'] > 0 then begin'+NL + indent+'
';
     end else begin
     end;


       t:=trim(Xls.GetStringFromCell(r, 5));
       ExtractNames(t);
       if t<>'' then
         tmps := tmps + '_Cost['+ci.ToString+'] := '+ t + ';'+NL + indent +
                 '  inc(TotEval);'+NL + indent
       else
         tmps := tmps + '_Cost['+ci.ToString+'] := 0;'+NL + indent +
                'inc(TotEval);'+NL + indent
         ;
       if usecc then tmps := tmps + '  ';


       t:=trim(Xls.GetStringFromCell(r, 8));
       ExtractNames(t);
       if t<>'' then
         tmps := tmps + '_OM['+ci.ToString+'] := '+ t + ';'+NL + indent +
                '  inc(TotEval);'+NL + indent
       else
         tmps := tmps + '_OM['+ci.ToString+'] := 0;'+NL + indent +
                'inc(TotEval);'+NL + indent;

       if usecc then tmps := tmps + 'end;'+NL + indent;
     inc(ci);

     if aState then
       evstringSt := tmps
     else
       evstring := tmps;
  end;
  FreeAndNil(Xls);
  if ci>MaxR then MaxR:=ci;
  if fBook = 'BASELINE' then
    tmpString := StringReplace(tmpString,'9999', ci.ToString,
[rfIgnoreCase,rfReplaceAll])
```

```
  else
    tmpString := StringReplace(tmpString,'9998', ci.ToString,
[rfIgnoreCase,rfReplaceAll]);
  tmpString := StringReplace(tmpString,'(*SetState'+fBook+'*)', SetState,
[rfIgnoreCase,rfReplaceAll]);
  tmpString := StringReplace(tmpString,'(*EVALUATE'+fBook+'*)',evString,
[rfIgnoreCase,rfReplaceAll]);
  tmpString := StringReplace(tmpString,'(*EVALUATESTATE'+fBook+'*)',evStringSt,
[rfIgnoreCase,rfReplaceAll]);
end;



end.
```