```
unit LCRCompiledCostTemplate;

//Autogenerated unit -
//Generated - (*DATE*)
//Baseline Workbook - (*WORKBOOKBASELINE*)
//Found variables in workbook - (*NUMVARSBASELINE*)
//Found costing steps - 9999
//Option Workbook - (*WORKBOOKOPTION*)
//Found variables in workbook - (*NUMVARSOPTION*)
//Found costing steps - 9998

interface

uses Math, SysUtils;

type
  TLSRValueStore = array[0..9997] of double;
  TLSRValueStoreFlag = array[0..9997] of boolean;
  TLSRCalcYearStore = array[0..100,0..9997] of boolean;

  _VariablesRecBASELINE = record
(*VARIABLESBASELINE*)
  end;
  _VariablesRecOPTION = record
(*VARIABLESOPTION*)
  end;

  TLSRCompiledCost = class
    _CalcCost,_Cost,_Hours,_OM,_Labor  : TLSRValueStore;
    _ImAState : TLSRValueStoreFlag;
    _YearOK : TLSRCalcYearStore;
    TotEval : int64;
    constructor create();
    procedure _Reset(aState : boolean);
    procedure _LoadVars; virtual; abstract;
    procedure _SetVarPointer(const name : string; const pd : PDouble); virtual;
abstract;
    procedure _Evaluate(const Yr : integer); virtual; abstract;
    procedure _EvaluateState(const Yr: integer); virtual; abstract;
    function _DumpVars : string; virtual; abstract;

  end;

  TLSRCompiledCostBaseline = class(TLSRCompiledCost)
    _Variables : _VariablesRecBaseline;

    constructor create();
    procedure _LoadVars; override;
    procedure _SetVarPointer(const name : string; const pd : PDouble); override;
    procedure _Evaluate(const Yr : integer); override;
    procedure _EvaluateState(const Yr: integer); override;

    function _DumpVars : string; override;
  end;

  TLSRCompiledCostOption = class(TLSRCompiledCost)
    _Variables : _VariablesRecOption;

    constructor create();
    procedure _LoadVars; override;
    procedure _SetVarPointer(const name : string; const pd : PDouble); override;
    procedure _Evaluate(const Yr : integer); override;
    procedure _EvaluateState(const Yr: integer); override;

    function _DumpVars : string; override;
  end;


const
  _CWorkBookBASELINE = '(*WORKBOOKBASELINE*)';
  _CWorkBookOPTION = '(*WORKBOOKOPTION*)';
  _CWorkBookDate = '(*DATE*)';

var
  _UseCompiled : boolean = false;

implementation

{ TLSRCompiledCost }

constructor TLSRCompiledCost.create;
begin
  TotEval:=0;
end;

procedure TLSRCompiledCost._Reset(aState : boolean);
var i : integer;
begin
  for i:=0 to high(_CalcCost) do begin
    if not (_ImAState[i] = aState) then continue;
    _CalcCost[i]:=0;
    _Cost[i]:=0;
    _Hours[i]:=0;
    _OM[i]:=0;
    _Labor[i]:=0;
  end;
```

```
end;

{ TLSRCompiledCostBaseline }

procedure TLSRCompiledCostBaseline._LoadVars;
begin
(*LoadVarsBASELINE*)
end;

procedure TLSRCompiledCostBaseline._Evaluate(const Yr : integer);
begin
  _Reset(False);
  _LoadVars();
  with _Variables do begin
(*EVALUATEBASELINE*)
  end;
end;

procedure TLSRCompiledCostBaseline._EvaluateState(const Yr : integer);
begin
  _Reset(True);
  _LoadVars();
  with _Variables do begin
(*EVALUATESTATEBASELINE*)
  end;
end;


procedure TLSRCompiledCostBaseline._SetVarPointer(const name : string; const pd :
PDouble);
var s : string;
begin
  s := lowercase(name);
(*_SetVarPointerBASELINE*)
end;

function TLSRCompiledCostBaseline._DumpVars: string;
var s : string;
begin
  s:='';
(*DumpVarsBASELINE*)
  result:=s;
end;

constructor TLSRCompiledCostBaseline.create;
begin
  inherited create;
(*SetStateBASELINE*)
```

```
end;

{ TLSRCompiledCostOption }

procedure TLSRCompiledCostOption._LoadVars;
begin
(*LoadVarsOPTION*)
end;

procedure TLSRCompiledCostOption._Evaluate(const Yr : integer);
begin
  _Reset(False);
  _LoadVars();
  with _Variables do begin
(*EVALUATEOPTION*)
  end;
end;

procedure TLSRCompiledCostOption._EvaluateState(const Yr : integer);
begin
  _Reset(True);
  _LoadVars();
  with _Variables do begin
(*EVALUATESTATEOPTION*)
  end;
end;


procedure TLSRCompiledCostOption._SetVarPointer(const name : string; const pd :
PDouble);
var s : string;
begin
  s := lowercase(name);
(*_SetVarPointerOPTION*)
end;

function TLSRCompiledCostOption._DumpVars: string;
var s : string;
begin
  s:='';
(*DumpVarsOPTION*)
  result:=s;
end;

constructor TLSRCompiledCostOption.create;
begin
  inherited create;
(*SetStateOPTION*)
```

```
end;


end.
```