Балтийский государственный технический университет «Военмех» им. Д. Ф. Устинова

Кафедра И5

«Информационные системы и программная инженерия»

**ЛАБОРАТОРНАЯ РАБОТА № 5**

По дисциплине **«Визуальное программирование»**

На тему

**«ПОТОКИ. СЕРИАЛИЗАЦИЯ. КОМПОНЕНТ TREEVIEW»**

***Вариант № 4***

Выполнил:

Студент Васильев Н. А.

Группа И967

**Преподаватель**:

Ракова И. К.

Санкт-Петербург

2018

**Цель работы**:

Научиться эффективно использовать потоки; применять компонент TreeView; освоить сериализацию.

**Задание:**

Необходимо разработать приложение, позволяющее манипулировать древовидными структурами данных (n-арными деревьями).

Реализовать приложение, отображающее дерево каталогов и графических файлов с расширениями \*.jpg, \*.bmp и позволяющее просматривать их с помощью редактора по умолчанию, зарегистрированного в ОС Windows для данных файлов. При запуске программы пользователь должен в диалоговом окне указать путь к каталогу, содержимое которого будет в дальнейшем считано. Отображение структуры каталогов и файлов должно производиться с использованием компонентов ListView и TreeView. При нажатии на кнопку “Сохранить” дерево каталогов должно быть сохранено в файл, при нажатии на кнопку “Загрузить” – восстановлено из файла.

**Текст программы:**

**nodeunit**:

unit nodeunit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils;

type

TFileNode = class(TCollectionItem)

private

fileName: string;

parent: integer;

public

constructor Create;

published

property Pname: string read fileName write fileName;

property Pparent: integer read parent write parent;

end;

implementation

constructor TFileNode.Create;

begin

end;

end.

**treeunit:**

unit treeunit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, nodeunit, Dialogs, ComCtrls;

type

TFileTree = class(TComponent)

private

files: TCollection;

treeView: TTreeView;

Ckk: integer;

startFolder: string;

function GetCount: Integer;

function Get(index: Integer): TFileNode;

public

constructor Create;

destructor Destroy; override;

procedure Add(nodeName: String; Parent: Integer);

procedure addByIndex(index: Integer; nodeName: String; Parent: Integer);

procedure InsertNode(index: Integer; nodeName: String; Parent: Integer);

procedure syncNodesWithTreeView();

procedure Delete(index: Integer);

function CountOfNodes(node: TTreeNode): Integer;

function IndexOfLastNode(node: TTreeNode): Integer;

procedure SaveToStream(stream: TStream);

procedure ReadFromStream(stream: TStream);

procedure Sync();

procedure Clear();

procedure Link(newnode: TTreeNode; number: Integer);

property PItems[index: Integer]: TFileNode read Get;

property PCount: Integer read GetCount;

property PTreeView: TTreeView write treeView;

property StartFolder1: String read StartFolder write StartFolder;

property CkkItem1: Integer read Ckk write Ckk;

published

property Items: TCollection read files write files;

property StartFolderString: String read startFolder write startFolder;

property CkkItem: Integer read Ckk write Ckk;

end;

implementation

procedure TFileTree.Clear();

begin

files.Clear;

treeView.Items.Clear;

end;

procedure TFileTree.addByIndex(index:Integer; nodeName:String;Parent:Integer);

begin

if index < files.Count then

InsertNode(index, nodeName, Parent)

else

Add(nodeName, Parent);

end;

procedure TFileTree.Add(nodeName: String; Parent: Integer);

begin

files.Add;

TFileNode(files.Items[files.Count-1]).PName:=nodeName;

TFileNode(files.Items[files.Count-1]).PParent:=Parent;

end;

procedure TFileTree.InsertNode(index: Integer; nodeName: String; Parent: Integer);

begin

files.Insert(index);

TFileNode(files.Items[Index]).PName:=nodeName;

TFileNode(files.Items[Index]).PParent:=Parent;

end;

function TFileTree.Get(index: Integer): TFileNode;

begin

Result:=TFileNode(files.Items[index]);

end;

constructor TFileTree.Create;

begin

files:=TCollection.Create(TFileNode);

end;

destructor TFileTree.Destroy;

begin

files.Destroy;

end;

procedure TFileTree.Sync();

var i, ilast: Integer;

begin

i:=0;

ilast:=files.Count;

while i < ilast do

begin

PItems[i].PName:=treeView.Items[i].Text;

inc(i);

end;

end;

procedure TFileTree.syncNodesWithTreeView();

var i, j: Integer;

begin

i:=0;

while i < files.Count do

begin

with TFileNode(files.Items[i]), treeView do

if PParent = -1 then

Items.AddChild(nil, PName)

else

Items.AddChild(Items[PParent], PName);

inc(i);

inc(j);

end;

end;

procedure TFileTree.Delete(index: Integer);

begin

files.Delete(index);

end;

function TFileTree.GetCount:Integer;

begin

Result:=files.Count;

end;

function TFileTree.CountOfNodes(node: TTreeNode): Integer;

begin

Result:=IndexOfLastNode(node)-(node.AbsoluteIndex)+1;

Ckk:=Result;

end;

function TFileTree.IndexOfLastNode(node: TTreeNode): Integer;

var

i, level: Integer;

begin

i:=node.AbsoluteIndex;

level:=node.Level;

repeat

inc(i);

if i=treeView.Items.Count then

break;

until

TreeView.Items[i].Level<=level;

Result:=i-1;

end;

procedure TFileTree.SaveToStream(stream: TStream);

var

i, j, l: Integer;

node: TFileNode;

begin

i:=0;

j:=TreeView.Items[0].AbsoluteIndex;

while i<Ckk do

begin

node:=TFileNode(files.Items[j]);

l:=Length(node.PName);

stream.Write(l, sizeof(integer));

stream.WriteBuffer(node.PName[1], l);

stream.Write(node.PParent, sizeof(Integer));

inc(j);

inc(i);

end;

end;

procedure TFileTree.Link(newnode: TTreeNode; number: Integer);

var

i, d, p: Integer;

begin

with newnode do

begin

i:=AbsoluteIndex;

if number=1 then

begin

if Level>0 then

TFileNode(files.Items[i]).PParent:=Parent.AbsoluteIndex

else

TFileNode(files.Items[i]).PParent:=-1;

end else

if number>1 then

begin

p:=TFileNode(files.Items[i+1]).PParent;

if Level>0 then

TFileNode(files.Items[i]).PParent:=Parent.AbsoluteIndex

else

TFileNode(files.Items[i]).PParent:=-1;

TFileNode(files.Items[i+1]).PParent:=i;

inc(i,2);

dec(number,2);

while number>0 do

begin

d:=TFileNode(files.Items[i]).PParent-p;

p:=TFileNode(files.Items[i]).PParent;

TFileNode(files.Items[i]).PParent:=TFileNode(files.Items[i-1]).PParent+d;

inc(i,1);

dec(number,1);

end;

end;

end;

end;

procedure TFileTree.ReadFromStream(stream: TStream);

var

l, i, j, c, parent: Integer;

name1: String;

node: TFileNode;

begin

if Ckk>0 then

begin

c:=Ckk;

node:=TFileNode.Create;

stream.Read(l, sizeof(Integer));

setlength(name1, l);

stream.ReadBuffer(name1[1], l);

node.PName:=name1;

stream.Read(parent, sizeof(Integer));

node.PParent:=parent;

with TreeView do

i:=Items.AddChild(Selected, node.PName).AbsoluteIndex;

if i=files.Count then

begin

Add(node.PName, node.PParent);

dec(Ckk);

while Ckk>0 do

begin

node:=TFileNode.Create;

stream.Read(l, sizeof(Integer));

setlength(name1, l);

stream.ReadBuffer(name1[1], l);

node.PName:=name1;

stream.Read(parent, sizeof(Integer));

node.PParent:=parent;

Add(node.PName, node.PParent);

dec(Ckk);

end;

end else

begin

InsertNode(i, node.PName, node.PParent);

dec(Ckk);

j:=i+1;

while Ckk>0 do

begin

node:=TFileNode.Create;

stream.Read(l, sizeof(Integer));

setlength(name1, l);

stream.ReadBuffer(name1[1], l);

node.PName:=name1;

stream.Read(parent, sizeof(Integer));

node.PParent:=parent;

InsertNode(j, node.PName, node.PParent);

inc(j);

dec(Ckk);

end;

end;

Link(TreeView.Items[i], c);

j:=1;

inc(i);

while j<c do

begin

with TFileNode(files.Items[i]),TreeView do

Items.AddChild(Items[PParent], PName);

inc(i);

inc(j);

end;

end;

end;

end.

**syncunit:**

unit syncunit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, ComCtrls, treeunit, nodeunit;

type

TSynchronizer = class(TObject)

public

procedure Add(tree: TFileTree; node: TFileNode; index: Integer);

procedure Edit(tree: TFileTree; index: Integer; s: String);

procedure Delete(tree: TFileTree; node: TTreeNode);

end;

implementation

procedure TSynchronizer.Add(tree: TFileTree; node: TFileNode; index: Integer);

begin

tree.addByIndex(index, node.PName, node.PParent);

end;

procedure TSynchronizer.Edit(tree: TFileTree; index: Integer; s: String);

begin

tree.PItems[index].PName:=s;

end;

procedure TSynchronizer.Delete(tree: TFileTree; node: TTreeNode);

var

i, ilast: Integer;

begin

i:=node.AbsoluteIndex;

ilast:=tree.IndexOfLastNode(node);

while ilast>=i do

begin

tree.Delete(ilast);

dec(ilast);

end;

end;

end.

**Модуль формы Unit1:**

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,

ComCtrls, Menus, treeunit, syncunit, nodeunit, LazFileUtils, LCLIntf ;

type

{ TForm1 }

TForm1 = class(TForm)

ClearBtn: TButton;

Label1: TLabel;

Label2: TLabel;

SaveH: TButton;

SaveL: TButton;

RestorH: TButton;

RestorL: TButton;

openFolderButton: TButton;

SelectDirectory: TSelectDirectoryDialog;

TreeView1: TTreeView;

procedure ClearBtnClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure openFolderButtonClick(Sender: TObject);

procedure RestorHClick(Sender: TObject);

procedure RestorLClick(Sender: TObject);

procedure SaveHClick(Sender: TObject);

procedure SaveLClick(Sender: TObject);

procedure TreeView1DblClick(Sender: TObject);

public

procedure GetDir(ParentNode: TTreeNode);

function GetPathRec(Node: TTreeNode):string;

end;

var

Form1: TForm1;

StreamT, StreamBL, StreamBH: TStream;

Node: TFileNode;

Tree: TFileTree;

Sync: TSynchronizer;

ser\_count: Integer;

startFolder: String;

implementation

{$R \*.lfm}

{ TForm1 }

procedure TForm1.openFolderButtonClick(Sender: TObject);

var node: TFileNode;

TreeTNode: TTreeNode;

begin

if SelectDirectory.Execute then

begin

Tree.Clear;

node:= TFileNode.Create;

TreeView1.Items.Clear;

startFolder := SelectDirectory.FileName;

TreeTNode := TreeView1.Items.AddChild(nil, startFolder);

node.PName:=startFolder;

node.PParent:=-1;

Sync.Add(Tree, node, TreeTNode.AbsoluteIndex);

GetDir(TreeTNode);

end;

end;

procedure TForm1.RestorHClick(Sender: TObject);

var MemStream: TMemoryStream;

begin

Tree.Clear;

StreamBH:=TFileStream.Create(extractfilepath(Application.ExeName) + 'fileH.dat', fmOpenRead);

MemStream:=TMemoryStream.Create;

RegisterClass(TFileTree);

RegisterClass(TCollection);

RegisterClass(TFileNode);

ObjectTextToBinary(StreamBH, MemStream);

MemStream.Position:=0;

MemStream.ReadComponent(Tree);

Tree.syncNodesWithTreeView;

StartFolder:=Tree.StartFolder1;

MemStream.Free;

StreamBH.Free;

end;

procedure TForm1.RestorLClick(Sender: TObject);

begin

Tree.Clear;

StreamBL:=TFileStream.Create(extractfilepath(Application.ExeName)+'fileL.dat', fmOpenRead);

Tree.ReadFromStream(StreamBL);

StreamBL.Free;

end;

procedure TForm1.SaveHClick(Sender: TObject);

var MemStream: TMemoryStream;

begin

if TreeView1.Items.Count<>0 then

begin

Tree.Sync();

Tree.StartFolder1:=StartFolder;

StreamBH:=TFileStream.Create(extractfilepath(Application.ExeName)+'fileH.dat', fmCreate);

MemStream:=TMemoryStream.Create;

MemStream.WriteComponent(Tree);

MemStream.Position:=0;

ObjectBinaryToText(MemStream, StreamBH);

StreamBH.Free;

MemStream.Free;

end;

end;

procedure TForm1.SaveLClick(Sender: TObject);

begin

if TreeView1.Selected <> nil then

begin

if TreeView1.Items.Count<>0 then

begin

Tree.StartFolder1:=StartFolder;

Tree.Sync();

StreamBL:=TFileStream.Create(extractfilepath(Application.ExeName)+'fileL.dat', fmCreate);

StreamBL.Seek(0,0);

Tree.CountOfNodes(TreeView1.Items[0]);

Tree.SaveToStream(StreamBL);

StreamBL.Free;

end;

end;

end;

procedure TForm1.TreeView1DblClick(Sender: TObject);

var

temp: string;

begin

temp:='';

if TreeView1.Selected <> nil then

begin

temp:=TreeView1.Selected.Text;

if((Pos('jpg',temp) <> 0) or (Pos('bmp',temp) <> 0) or (Pos('JPG',temp) <> 0) or (Pos('BMP',temp) <> 0))then

begin

OpenDocument(GetPathRec(TreeView1.Selected));

end;

end;

temp:='';

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

Tree:=TFileTree.Create;

Sync:=TSynchronizer.Create;

Tree.PTreeView:=TreeView1;

end;

procedure TForm1.ClearBtnClick(Sender: TObject);

begin

Tree.Clear;

end;

procedure TForm1.GetDir(ParentNode: TTreeNode);

var

sr: TSearchRec;

GetDirNode: TTreeNode;

node: TFileNode;

path: string;

i: integer;

begin

node:=TFileNode.Create;

GetDirNode := ParentNode;

path := '';

i:=1;

repeat

path := IncludeTrailingPathDelimiter(GetDirNode.Text) + path;

GetDirNode := GetDirNode.Parent;

until GetDirNode = nil;

if FindFirstUTF8(path + '\*.\*', faDirectory, sr) = 0 then

try

repeat

if (sr.Name = '.') or (sr.Name = '..') or (sr.Attr and faDirectory <> faDirectory) then Continue;

GetDirNode := TreeView1.Items.AddChild(ParentNode, sr.Name);

if i=1 then

begin

node.PName:=sr.Name;

node.PParent:=ParentNode.AbsoluteIndex;

Sync.Add(Tree, node, GetDirNode.AbsoluteIndex);

end

else

begin

node.PName:=sr.Name;

node.PParent:=ParentNode.AbsoluteIndex;

Sync.Add(Tree, node, GetDirNode.AbsoluteIndex);

end;

GetDir(GetDirNode);

inc(i);

until FindNextUTF8(sr) <> 0;

finally

FindCloseUTF8(sr);

end;

if (FindFirstUTF8(path + '\*.jpg', faAnyFile, sr) = 0) then

try

repeat

if (sr.Name = '.') or (sr.Name = '..') or (sr.Attr and faDirectory = faDirectory) then Continue;

node.PName:=sr.Name;

node.PParent:=ParentNode.AbsoluteIndex;

Sync.Add(Tree, node, Form1.TreeView1.Items.AddChild(ParentNode, sr.Name).AbsoluteIndex);

inc(i);

until FindNextUTF8(sr) <> 0;

finally

FindCloseUTF8(sr);

end;

if (FindFirstUTF8(path + '\*.JPG', faAnyFile, sr) = 0) then

try

repeat

if (sr.Name = '.') or (sr.Name = '..') or (sr.Attr and faDirectory = faDirectory) then Continue;

node.PName:=sr.Name;

node.PParent:=ParentNode.AbsoluteIndex;

Sync.Add(Tree, node, Form1.TreeView1.Items.AddChild(ParentNode, sr.Name).AbsoluteIndex);

inc(i);

until FindNextUTF8(sr) <> 0;

finally

FindCloseUTF8(sr);

end;

if (FindFirstUTF8(path + '\*.bmp', faAnyFile, sr) = 0) then

try

repeat

if (sr.Name = '.') or (sr.Name = '..') or (sr.Attr and faDirectory = faDirectory) then Continue;

node.PName:=sr.Name;

node.PParent:=ParentNode.AbsoluteIndex;

Sync.Add(Tree, node, Form1.TreeView1.Items.AddChild(ParentNode, sr.Name).AbsoluteIndex);

inc(i);

until FindNextUTF8(sr) <> 0;

finally

FindCloseUTF8(sr);

end;

if (FindFirstUTF8(path + '\*.BMP', faAnyFile, sr) = 0) then

try

repeat

if (sr.Name = '.') or (sr.Name = '..') or (sr.Attr and faDirectory = faDirectory) then Continue;

node.PName:=sr.Name;

node.PParent:=ParentNode.AbsoluteIndex;

Sync.Add(Tree, node, Form1.TreeView1.Items.AddChild(ParentNode, sr.Name).AbsoluteIndex);

inc(i);

until FindNextUTF8(sr) <> 0;

finally

FindCloseUTF8(sr);

end;

end;

function TForm1.GetPathRec(Node: TTreeNode): string;

var str:string;

begin

if Node.Parent.Text = StartFolder then

begin

GetPathRec:=(Node.Parent.Text+'\'+Node.Text);

end

else

begin

str:=GetPathRec(Node.Parent);

GetPathRec:=(str+'\'+Node.Text);

end;

end;

end.

**Результаты работы программы:**

