unit Unit12;

{$mode objfpc}{$H+}

interface

uses

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

ExtCtrls, Grids;

type

{ TOperForm }

TArray = Array [1..100] of String;

TOperForm = class(TForm)

MatrixA: TStringGrid;

MatrixB: TStringGrid;

MatrixC: TStringGrid;

aLoadFile: TButton;

bLoadFile: TButton;

aClear: TButton;

bClear: TButton;

cClear: TButton;

ExitBtn: TButton;

InfoPanel: TPanel;

Intersection: TImage;

Union: TImage;

ComplementA: TImage;

ComplementB: TImage;

MinusBA: TImage;

MinusAB: TImage;

SymMinus: TImage;

procedure ExitBtnClick(Sender: TObject);

procedure LoadFromFile(M:String);

procedure aLoadFileClick(Sender: TObject);

procedure bLoadFileClick(Sender: TObject);

procedure ClearGrid(M:String);

procedure aClearClick(Sender: TObject);

procedure bClearClick(Sender: TObject);

procedure cClearClick(Sender: TObject);

procedure addToArray(var arr: TArray; value:String; var count:Integer);

procedure Peretun(arrA:TArray; countA:Integer; arrB:TArray; countB:Integer; var arrC:TArray; var countC:Integer);

procedure Minus(arrA:TArray; countA:Integer; arrB:TArray; countB:Integer; var arrC:TArray; var countC:Integer);

procedure IntersectionClick(Sender: TObject);

procedure UnionClick(Sender: TObject);

procedure ComplementAClick(Sender: TObject);

procedure ComplementBClick(Sender: TObject);

procedure MinusABClick(Sender: TObject);

procedure MinusBAClick(Sender: TObject);

procedure SymMinusClick(Sender: TObject);

private

{ private declarations }

public

{ public declarations }

end;

Type TMatrix = Array [1..10,1..10] of String;

var

OperForm: TOperForm;

A0GridCol,A0GridRow,B0GridCol,B0GridRow, C0GridCol,C0GridRow:Integer;

A0Arr,B0Arr:TMatrix;

implementation

{$R \*.lfm}

Const MaxCol=10;

MaxRow=10;

Var LocalPath:String;

{ TOperForm }

procedure TOperForm.ExitBtnClick(Sender: TObject);

begin

Close;

end;

procedure TOperForm.LoadFromFile(M:String);

var F : Text;

LogFileName,Str:String;

begin

ClearGrid(M);

LogFileName:=LocalPath+'DATA\'+M+'.TXT';

AssignFile(F,LogFileName);

{$I-} Reset(F); {$I+}

if IOResult <> 0 then

begin

InfoPanel.Caption:='Impossibly to read from '+LogFileName;

Exit;

end;

If M='A0' then

begin

A0GridCol:=0;

A0GridRow:=1;

While not EOF(F) do

begin

If A0GridCol<MaxCol then Inc(A0GridCol) else

begin

A0GridCol:=1;

Inc(A0GridRow);

If A0GridRow>MaxRow then

begin

InfoPanel.Caption:='Data from file А.TXT cant put in table';

CloseFile(F);

Exit;

end;

end;

Readln(F, Str); MatrixA.Cells[A0GridCol,A0GridRow]:=Str;

end;

end;

If M='B0' then

begin

B0GridCol:=0;

B0GridRow:=1;

While not EOF(F) do

begin

If B0GridCol<MaxCol then Inc(B0GridCol) else

begin

B0GridCol:=1;

Inc(B0GridRow);

If A0GridRow>MaxRow then

begin

InfoPanel.Caption:='Data from file B.TXT cant put in table';

CloseFile(F);

Exit;

end;

end;

Readln(F, Str); MatrixB.Cells[B0GridCol,B0GridRow]:=Str;

end;

end;

CloseFile(F);

end;

procedure TOperForm.aLoadFileClick(Sender: TObject);

begin

LoadFromFile('A0');

end;

procedure TOperForm.bLoadFileClick(Sender: TObject);

begin

LoadFromFile('B0');

end;

procedure TOperForm.ClearGrid(M:String);

var i,j:Integer;

begin

If M='A0' then begin

For i:=1 to MaxCol do

For j:=1 to MaxRow do MatrixA.Cells[i,j]:='';

A0GridCol:=0;

A0GridRow:=1;

end;

If M='B0' then begin

For i:=1 to MaxCol do

For j:=1 to MaxRow do MatrixB.Cells[i,j]:='';

B0GridCol:=0;

B0GridRow:=1;

end;

If M='C0' then begin

For i:=1 to MaxCol do

For j:=1 to MaxRow do MatrixC.Cells[i,j]:='';

C0GridCol:=0;

C0GridRow:=1;

end;

end;

procedure TOperForm.aClearClick(Sender: TObject);

begin

ClearGrid('A0');

end;

procedure TOperForm.bClearClick(Sender: TObject);

begin

ClearGrid('B0');

end;

procedure TOperForm.cClearClick(Sender: TObject);

begin

ClearGrid('C0');

end;

procedure TOperForm.addToArray(var arr: TArray; value:String; var count:Integer);

var i : Integer;

q : Boolean;

begin

q := true;

for i := 1 to count do

if (arr[i] = value) then begin

q := false;

break;

end;

if (q) then begin

INC(count);

arr[count] := value;

end;

end;

procedure TOperForm.Peretun(arrA:TArray; countA:Integer; arrB:TArray; countB:Integer; var arrC:TArray; var countC:Integer);

var

i,j:Integer;

begin

countC:=0;

for i := 1 to countA do

for j := 1 to countB do

if( arrA[i] = arrB[j]) then

begin

INC(countC);

arrC[countC] := arrA[i];

end;

end;

procedure TOperForm.Minus(arrA:TArray; countA:Integer; arrB:TArray; countB:Integer; var arrC:TArray; var countC:Integer);

var

i,j:Integer;

q:boolean;

begin

countC:=0;

for i := 1 to countA do

begin

q := true;

for j := 1 to countB do

if( arrA[i] = arrB[j]) then

q :=false;

if(q) then

begin

INC(countC);

arrC[countC] := arrA[i];

end;

end;

end;

procedure TOperForm.IntersectionClick(Sender: TObject);

var

i,j:Integer;

arrA, arrB, arrC:TArray;

countA, countB, countC:Integer;

begin

countA :=0;

countB :=0;

ClearGrid('C0');

For j:=1 to 10 do

For i:=1 to 10 do

begin

if (MatrixA.Cells[i, j] <> '') then

addToArray(arrA, MatrixA.Cells[i, j], countA);

if (MatrixB.Cells[i, j] <> '') then

addToArray(arrB, MatrixB.Cells[i, j], countB);

end;

Peretun(arrA, countA, arrB, countB, arrC, countC);

i := 0;

while ((i < countC) AND (i < 10\*10)) do begin

MatrixC.Cells[(i MOD 10) + 1, (i DIV 10) + 1] := arrC[i + 1];

INC(i);

end;

InfoPanel.Caption := 'Intersection sets A&B';

end;

procedure TOperForm.UnionClick(Sender: TObject);

var

i,j,count : Integer;

arr : TArray;

begin

count := 0;

ClearGrid('C0');

For j:=1 to 10 do

For i:=1 to 10 do

if (MatrixA.Cells[i, j] <> '') then

addToArray(arr, MatrixA.Cells[i, j], count);

{MatrixC.Cells[i,j] := '1'MatrixA.Cells[i,j]};

For j:=1{A0GridCol} to 10{B0GridCol+A0GridCol} do

For i:=1{A0GridRow} to 10{B0GridRow+a0GridRow} do

if (MatrixB.Cells[i, j] <> '') then

addToArray(arr, MatrixB.Cells[i, j], count);

{MatrixC.Cells[i,j] := MatrixB.Cells[i-A0GridCol, j-A0GridRow];}

{For i:=0 to MaxCol-1 do

For j:=1 to MaxRow do MatrixC.Cells[i+1,j]:=arr[i \* 10 + j];}

i := 0;

while ((i < count) AND (i < 10\*10)) do begin

MatrixC.Cells[(i MOD 10) + 1, (i DIV 10) + 1] := arr[i + 1];

INC(i);

end;

InfoPanel.Caption := 'Union sets A&B';

end;

procedure TOperForm.ComplementAClick(Sender: TObject);

var

i,j:Integer;

AllArr, loadedArr, result:TArray;

countA, countB, countC:Integer;

begin

ClearGrid('C0');

For i:=1 to MaxRow do

For j:=1 to MaxCol do

begin

AllArr[(j-1)\*10 + i]:=Chr(19+j\*10+i);

end;

countA := 10\*10;

countB:=0;

For j:=1 to 10 do

For i:=1 to 10 do

begin

if (MatrixA.Cells[i, j] <> '') then

addToArray(loadedArr, MatrixA.Cells[i, j], countB);

end;

minus(AllArr, countA, loadedArr, countB, result, countC);

i := 0;

while ((i < countC) AND (i < 10\*10)) do begin

MatrixC.Cells[(i MOD 10) + 1, (i DIV 10) + 1] := result[i + 1];

INC(i);

end;

InfoPanel.Caption := 'Complement a';

end;

procedure TOperForm.ComplementBClick(Sender: TObject);

var

i,j:Integer;

AllArr, loadedArr, result:TArray;

countA, countB, countC:Integer;

begin

ClearGrid('C0');

For i:=1 to MaxRow do

For j:=1 to MaxCol do

begin

AllArr[(j-1)\*10 + i]:=Chr(19+j\*10+i);

end;

countA := 10\*10;

countB:=0;

For j:=1 to 10 do

For i:=1 to 10 do

begin

if (MatrixB.Cells[i, j] <> '') then

addToArray(loadedArr, MatrixB.Cells[i, j], countB);

end;

minus(AllArr, countA, loadedArr, countB, result, countC);

i := 0;

while ((i < countC) AND (i < 10\*10)) do begin

MatrixC.Cells[(i MOD 10) + 1, (i DIV 10) + 1] := result[i + 1];

INC(i);

end;

InfoPanel.Caption := 'Complement b';

end;

procedure TOperForm.MinusABClick(Sender: TObject);

var

i,j:Integer;

arrA, arrB, arrC:TArray;

countA, countB, countC:Integer;

begin

countA :=0;

countB :=0;

ClearGrid('C0');

For j:=1 to 10 do

For i:=1 to 10 do

begin

if (MatrixA.Cells[i, j] <> '') then

addToArray(arrA, MatrixA.Cells[i, j], countA);

if (MatrixB.Cells[i, j] <> '') then

addToArray(arrB, MatrixB.Cells[i, j], countB);

end;

minus(arrA, countA, arrB, countB, arrC, countC);

i := 0;

while ((i < countC) AND (i < 10\*10)) do begin

MatrixC.Cells[(i MOD 10) + 1, (i DIV 10) + 1] := arrC[i + 1];

INC(i);

end;

InfoPanel.Caption := 'Minus A-B';

end;

procedure TOperForm.MinusBAClick(Sender: TObject);

var

i,j:Integer;

arrA, arrB, arrC:TArray;

countA, countB, countC:Integer;

begin

countA :=0;

countB :=0;

ClearGrid('C0');

For j:=1 to 10 do

For i:=1 to 10 do

begin

if (MatrixA.Cells[i, j] <> '') then

addToArray(arrA, MatrixA.Cells[i, j], countA);

if (MatrixB.Cells[i, j] <> '') then

addToArray(arrB, MatrixB.Cells[i, j], countB);

end;

minus(arrB, countB, arrA, countA, arrC, countC);

i := 0;

while ((i < countC) AND (i < 10\*10)) do begin

MatrixC.Cells[(i MOD 10) + 1, (i DIV 10) + 1] := arrC[i + 1];

INC(i);

end;

InfoPanel.Caption := 'Minus B-A';

end;

procedure TOperForm.SymMinusClick(Sender: TObject);

var

i,j:Integer;

arrA, arrB, arrC1, arrC2:TArray;

countA, countB, countC1, countC2:Integer;

begin

countA :=0;

countB :=0;

ClearGrid('C0');

For j:=1 to 10 do

For i:=1 to 10 do

begin

if (MatrixA.Cells[i, j] <> '') then

addToArray(arrA, MatrixA.Cells[i, j], countA);

if (MatrixB.Cells[i, j] <> '') then

addToArray(arrB, MatrixB.Cells[i, j], countB);

end;

minus(arrA, countA, arrB, countB, arrC1, countC1);

minus(arrB, countB, arrA, countA, arrC2, countC2);

for i:=1 to countC2 do

begin

INC(countC1);

arrC1[countC1] := arrC2[i];

end;

i := 0;

while ((i < countC1) AND (i < 10\*10)) do begin

MatrixC.Cells[(i MOD 10) + 1, (i DIV 10) + 1] := arrC1[i + 1];

INC(i);

end;

InfoPanel.Caption := 'Symmetric minus of sets A&B';

end;

end.