УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №6.2

по предмету

Основы алгоритмизации и программирования

Вариант 14

Выполнил:

Крутько А.А.

Проверила:

Данилова Г.В.

Группа 251004

Минск 2023

Задание:



Код программы на **Delphi**:

unit Unit1;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Vcl.StdCtrls, Vcl.Buttons,

Vcl.ExtCtrls, ClipBRD, Vcl.Menus;

type

TArr = Array of Array of Integer;

TLabeledEdit = Class(Vcl.ExtCtrls.TLabeledEdit)

Public

Procedure WMPaste(Var Msg:TMessage); Message WM\_PASTE;

End;

TMainForm = class(TForm)

NumberEdit: TLabeledEdit;

CalcButton: TBitBtn;

FirstSquare: TStringGrid;

SecondSquare: TStringGrid;

OpenFile: TOpenDialog;

SaveFile: TSaveDialog;

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

procedure FormCreate(Sender: TObject);

procedure NumberEditKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure NumberEditKeyPress(Sender: TObject; var Key: Char);

procedure NumberEditChange(Sender: TObject);

procedure CalcButtonClick(Sender: TObject);

procedure FirstSquareDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

procedure N4Click(Sender: TObject);

procedure N5Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

MainForm: TMainForm;

Arr: TArr;

implementation

{$R \*.dfm}

Procedure GetNumbersInTheFirst(Numb: Integer);

Begin

MainForm.FirstSquare.ColCount := Numb;

MainForm.FirstSquare.RowCount := Numb;

Var Count := 1;

For Var I := 0 to Numb - 1 do

For Var J := 0 to Numb - 1 do

Begin

MainForm.FirstSquare.Cells[J, I] := IntToStr(Count);

Inc(Count);

End;

MainForm.FirstSquare.Height := Numb \* (MainForm.FirstSquare.DefaultRowHeight + 2);

MainForm.FirstSquare.Width := Numb \* (MainForm.FirstSquare.DefaultColWidth + 2);

End;

Procedure PrintSecondSquare(Arr: TArr);

Begin

For Var I := 0 To MainForm.SecondSquare.RowCount - 1 do

For Var J := 0 To MainForm.SecondSquare.ColCount - 1 do

MainForm.SecondSquare.Cells[J, I] := IntToStr(Arr[I, J]);

End;

Function Odd(N: Integer): TArr;

Var

Arr: TArr;

Begin

SetLength(Arr, N, N);

Var I := N div 2;

Var J := N - 1;

Var Numb := 0;

While Numb < N \* N do

Begin

If (I = -1) and (J = N) Then

Begin

J := N - 2;

I := 0;

End

Else

Begin

If (J = N) Then

J := 0;

If (I < 0) Then

I := N - 1;

End;

If (Arr[I, J] = 0) Then

Arr[I,J] := Numb + 1

Else

Begin

Dec(J, 2);

Inc(I);

Continue;

End;

Inc(J);

Dec(I);

Inc(Numb);

End;

Result := Arr;

End;

Function DoublyEven(N: Integer): TArr;

Var

Arr: TArr;

I, J: Integer;

Begin

SetLength(Arr, N, N);

I := 0; J := 0;

While I < N do

Begin

J := 0;

While J < N Do

Begin

Arr[I,J] := N \* I + J + 1;

Inc(J);

End;

Inc(I);

End;

I := 0; J := 0;

While I < N Div 4 do

Begin

J := 0;

While J < N Div 4 Do

Begin

Arr[I,J] := (N \* N + 1) - Arr[I,J];

Inc(J);

End;

Inc(I);

End;

I := 0; J := 3 \* (N Div 4);

While I < N Div 4 do

Begin

J := 3 \* (N Div 4);

While J < N Do

Begin

Arr[I,J] := (N \* N + 1) - Arr[I,J];

Inc(J);

End;

Inc(I);

End;

I := 3 \* (N div 4); J := 0;

While I < N do

Begin

J := 0;

While J < N Div 4 Do

Begin

Arr[I,J] := (N \* N + 1) - Arr[I,J];

Inc(J);

End;

Inc(I);

End;

I := 3 \* (N div 4); J := 3 \* (N Div 4);

While I < N do

Begin

J := 3 \* (N Div 4);

While J < N Do

Begin

Arr[I,J] := (N \* N + 1) - Arr[I,J];

Inc(J);

End;

Inc(I);

End;

I := N div 4; J := N Div 4;

While I < 3 \* (N Div 4) do

Begin

J := N Div 4;

While J < 3 \* (N Div 4) Do

Begin

Arr[I,J] := (N \* N + 1) - Arr[I,J];

Inc(J);

End;

Inc(I);

End;

Result := Arr;

End;

Function SinglyEven(N: Integer): TArr;

Var

Arr: TArr;

SubSquare: TArr;

QuadrantFactors: Array [0..3] of Integer;

I, J: Integer;

Begin

SetLength(Arr, N, N);

Var Half := N div 2;

SubSquare := Odd(Half);

QuadrantFactors[0] := 0;

QuadrantFactors[1] := 2;

QuadrantFactors[2] := 3;

QuadrantFactors[3] := 1;

I := 0; J := 0;

While I < N do

Begin

J := 0;

While J < N do

Begin

Arr[I,J] := SubSquare[I Mod Half, J Mod Half] + QuadrantFactors[(I Div Half) \* 2 + J div Half] \* n \* n div 4;

Inc(J);

End;

Inc(I);

End;

Var NColsLeft := N Div 4;

Var NColsRight := NColsLeft - 1;

For I := 0 to N Div 2 - 1 do

Begin

For J := 0 to N - 1 do

If (J < NColsLeft) or (J >= N - NColsRight) or ((J = NColsLeft) and (I = NColsLeft)) Then

Begin

If (J = 0) and (I = nColsLeft) Then

Continue;

Var Temp := Arr[I,J];

Arr[I,J] := Arr[I + N div 2, J];

Arr[I + N div 2, J] := Temp;

End;

End;

Result := Arr;

End;

procedure TMainForm.CalcButtonClick(Sender: TObject);

Var

Arr: TArr;

begin

Var Count := SecondSquare.DefaultColWidth \* (StrToInt(NumberEdit.Text) - 3);

FirstSquare.Visible := True;

SecondSquare.Visible := True;

GetNumbersInTheFirst(StrToInt(NumberEdit.Text));

SecondSquare.ColCount := StrToInt(NumberEdit.Text);

SecondSquare.RowCount := StrToInt(NumberEdit.Text);

If StrToInt(NumberEdit.Text) > 3 Then

Begin

MainForm.Width := MainForm.Width + 2 \* ((StrToInt(NumberEdit.Text) - 3) \* (FirstSquare.DefaultColWidth + 2));

MainForm.Height := MainForm.Height + (StrToInt(NumberEdit.Text) - 3) \* FirstSquare.DefaultRowHeight;

SecondSquare.Left := SecondSquare.Left - Count;

NumberEdit.Left := NumberEdit.Left + Count;

CalcButton.Left := NumberEdit.Left;

End;

MainForm.SecondSquare.Height := StrToInt(NumberEdit.Text) \* (MainForm.SecondSquare.DefaultRowHeight + 2);

MainForm.SecondSquare.Width := StrToInt(NumberEdit.Text) \* (MainForm.SecondSquare.DefaultColWidth + 2);

MainForm.Position := PoDesktopCenter;

CalcButton.Enabled := False;

If (StrToInt(NumberEdit.Text)) Mod 2 = 1 Then

Arr := Odd(StrToInt(NumberEdit.Text));

If (StrToInt(NumberEdit.Text)) Mod 4 = 0 Then

Arr := DoublyEven(StrToInt(NumberEdit.Text));

If (StrToInt(NumberEdit.Text) - 2) Mod 4 = 0 Then

Arr := SinglyEven(StrToInt(NumberEdit.Text));

PrintSecondSquare(Arr);

N5.Enabled := True;

end;

procedure TMainForm.FirstSquareDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

begin

With TStringGrid(Sender) do

Begin

If gdSelected in state Then

begin

Canvas.Brush.Color:=clwhite;

Canvas.FillRect(Rect);

Canvas.Font:=Font;

DrawText(canvas.Handle, PChar(Cells[ACol, ARow]), -1, Rect,

DT\_NOPREFIX or DT\_SINGLELINE);

end;

End;

end;

procedure TMainForm.FormCreate(Sender: TObject);

begin

BorderIcons := BorderIcons - [biMaximize] + [biHelp];

BorderStyle := BsSingle;

Position := PoDesktopCenter;

end;

procedure TMainForm.N4Click(Sender: TObject);

Var

FileInput: TextFile;

Temp: String;

Size: Integer;

begin

If OpenFile.Execute Then

Begin

Try

Try

AssignFile(FileInput, OpenFile.FileName);

Reset(FileInput);

Readln(FileInput, Temp);

Size := StrToInt(Temp);

If (Size < 3) or (Size > 15) Then

raise Exception.Create('Проверьте правильность числа');

NumberEdit.Text := Temp;

N5.Enabled := True;

Except

on E : Exception do

Begin

ShowMessage('Поднята ошибка, с сообщением : ' + E.Message);

End;

End;

Finally

CloseFile(FileInput);

End;

End;

end;

procedure TMainForm.N5Click(Sender: TObject);

Var

FileOutput: TextFile;

begin

If SaveFile.Execute Then

Begin

Try

Try

AssignFile(FileOutput, ChangeFileExt(SaveFile.FileName, '.txt'));

ReWrite(FileOutput);

For Var I := 0 to SecondSquare.RowCount - 1 do

Begin

For Var J := 0 to SecondSquare.ColCount - 1 do

Write(FileOutput, SecondSquare.Cells[I, J]:4);

Writeln(FileOutput);

End;

//Тут вывод в файл

Except

On E: Exception do

ShowMessage('Проверьте файл, ошибка ' + E.Message);

End;

Finally

CloseFile(FileOutput);

End;

End;

end;

procedure TMainForm.NumberEditChange(Sender: TObject);

begin

CalcButton.Enabled := (Length(TEdit(Sender).Text) > 0) and (StrToInt(TEdit(Sender).Text) > 2);

FirstSquare.Visible := False;

SecondSquare.Visible := False;

MainForm.Height := 201;

MainForm.Width := 519;

NumberEdit.Left := 192;

CalcButton.Left := NumberEdit.Left;

FirstSquare.Left := 8;

SecondSquare.Left := 336;

N5.Enabled := False;

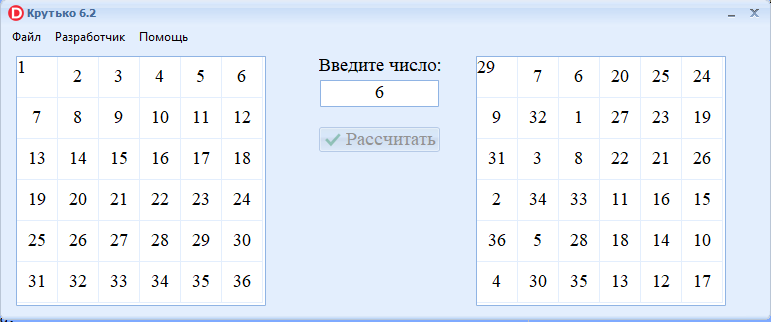
end;

end.

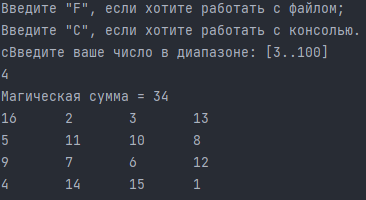
Код программы на **C#**:

ConsoleKeyInfo GetKey()  
{  
 ConsoleKeyInfo key;  
 Console.WriteLine("Введите \"F\", если хотите работать с файлом;\nВведите \"C\", если хотите работать с консолью.");  
 do {  
 key = Console.ReadKey();  
 } while (key.Key != ConsoleKey.C && key.Key != ConsoleKey.F);  
 return key;  
}  
  
int GetNumberC(in int min, in int max)  
{  
 Console.WriteLine("Введите ваше число в диапазоне: [3..100]");  
 int numb;  
 string? temp;  
 do {  
 temp = Console.ReadLine();  
 } while (!int.TryParse(temp, out numb) || numb < min || numb > max);  
 return numb;  
}  
  
void GetMagicSquareC(Action<object> action, in int min, in int max)  
{  
 var n = GetNumberC(min, max);  
 Console.WriteLine("Магическая сумма = " + n \* (n \* n + 1) / 2);  
 if (n % 2 == 1) Print(action, Odd(n));  
 if (n % 4 == 0) Print(action, DoublyEven(n));  
 if (n % 4 == 2) Print(action, SinglyEven(n));  
}  
  
int GetNumberF(in int min, in int max)  
{  
 bool isIncorrect;  
 int numb = 0;  
 do {  
 isIncorrect = true;  
 try  
 {  
 using (var st = new StreamReader(TakeFilePath()))  
 {  
 numb = int.Parse(st.ReadLine());  
 if (numb > max || numb < min)  
 throw new Exception ("Размер не соответствует валидному");  
 }  
 }  
 catch (Exception e)  
 {  
 Console.WriteLine(e.Message);  
 isIncorrect = false;  
 }  
 } while (!isIncorrect);  
 Console.WriteLine("Чтение из файла прошло успешно");  
 return numb;  
}  
  
string? TakeFilePath()  
{  
 string? path;  
 bool isIncorrect;  
 do {  
 isIncorrect = false;  
 Console.WriteLine("Введите путь к файлу");  
 path = Console.ReadLine();  
 if (!File.Exists(path) || !path.EndsWith(".txt"))  
 {  
 isIncorrect = true;  
 Console.WriteLine("Проверьте параметры файла");  
 }  
 } while (isIncorrect);  
 return path;  
}  
  
void SaveSquareToFile(int n)  
{  
 bool isIncorrect;  
 do {  
 isIncorrect = true;  
 try  
 {  
 using (var sw = new StreamWriter(TakeFilePath(), false))  
 {  
 sw.WriteLine("Магическая сумма = " + n \* (n \* n + 1) / 2);  
 if (n % 2 == 1)  
 Print(sw.Write, Odd(n));  
 if (n % 4 == 0)  
 Print(sw.Write, DoublyEven(n));  
 if (n % 4 == 2)  
 Print(sw.Write, SinglyEven(n));   
 }  
 }  
 catch (Exception e)  
 {  
 isIncorrect = false;  
 Console.WriteLine(e.Message);  
 }  
 } while (!isIncorrect);  
  
}  
  
void GetMagicSquareF(in int min, in int max)  
{  
 SaveSquareToFile(GetNumberF(min, max));  
}  
  
int[,] Odd(int n)  
{  
 int[,] arr = new int[n, n];  
 int i = n / 2;  
 int j = n - 1;  
 for (int num = 0; num < n \* n;)  
 {  
 if (i == -1 && j == n) {  
 j = n - 2;  
 i = 0;  
 }  
 else {  
 if (j == n) j = 0;  
 if (i < 0) i = n - 1;  
 }  
  
 if (arr[i, j] == 0)  
 arr[i, j] = ++num;  
 else {  
 j -= 2;  
 i++;  
 continue;   
 }  
 j++;  
 i--;  
 }  
 return arr;  
}  
  
int[,] DoublyEven(int n)  
{  
 int i, j;  
 int[,] arr = new int[n, n];  
  
 for (i = 0; i < n; i++)  
 for (j = 0; j < n; j++)  
 arr[i, j] = n \* i + j + 1;  
  
 for (i = 0; i < n / 4; i++)  
 for (j = 0; j < n / 4; j++)  
 arr[i, j] = (n \* n + 1) - arr[i, j];  
  
 for (i = 0; i < n / 4; i++)  
 for (j = 3 \* (n / 4); j < n; j++)  
 arr[i, j] = (n \* n + 1) - arr[i, j];  
  
 for (i = 3 \* n / 4; i < n; i++)  
 for (j = 0; j < n / 4; j++)  
 arr[i, j] = (n \* n + 1) - arr[i, j];  
  
 for (i = 3 \* n / 4; i < n; i++)  
 for (j = 3 \* n / 4; j < n; j++)   
 arr[i, j] = (n \* n + 1) - arr[i, j];  
  
 for (i = n / 4; i < 3 \* n / 4; i++)  
 for (j = n / 4; j < 3 \* n / 4; j++)  
 arr[i, j] = (n \* n + 1) - arr[i, j];  
 return arr;  
}  
  
int[,] SinglyEven(int n)  
{  
 int[,] arr = new int[n, n];  
 int half = n / 2;  
 int[,] subSquare = Odd(half);  
 int[] quadrantFactors = { 0, 2, 3, 1 };  
 for (int i = 0; i < n; i++)  
 for (int j = 0; j < n; j++)  
 arr[i, j] = subSquare[i % half, j % half] + quadrantFactors[i / half \* 2 + j / half] \* n \* n / 4;  
 int nColsLeft = n / 4;  
 int nColsRight = nColsLeft - 1;  
 for (int i = 0; i < n / 2; i++)  
 for (int j = 0; j < n; j++)  
 if (j < nColsLeft || j >= n - nColsRight || (j == nColsLeft && i == nColsLeft)) {  
 if (j == 0 && i == nColsLeft)  
 continue;  
 (arr[i, j], arr[i + n / 2, j]) = (arr[i + n / 2, j], arr[i, j]);  
 }  
 return arr;  
}  
  
void Print(Action<object> action, int[,] arr)  
{  
 for (int i = 0; i < arr.GetLength(0); i++) {  
 for (int j = 0; j < arr.GetLength(1); j++)  
 action(arr[i,j] + "\t");  
 action("\n");  
 }  
}  
  
if (GetKey().Key == ConsoleKey.C) GetMagicSquareC(Console.Write, 3, 20);  
else GetMagicSquareF(3, 20);

Результат на **Delphi**:



Результат на **C#**:



**Блок-cхема**:

