Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра програмування та захисту інформації

Звіт

з виконаної лабораторної роботи № 1

дисципліни “ Технології розробки алгоритмів ”

на тему

“ Аналіз часу виконання програми ”

Виконав

студент академічної

групи КІ-15

Аннаев А

Перевірив

Викл.

Гермак В.С.

Кропивницкий- 2017

Лабораторная работа №1

Тема: Анализ времени выполнения программы

Цель: Овладеть навыками определения времени и порядка степени роста алгоритма, приобрести навыки в разработке блок-схем, псевдокод алгоритмов и написании программ с использованием ветвлений и циклов

ход работы

Задание:

В соответствии со своим вариантом написать блок-схемы, псевдокод алгоритмов и программы для вычисления указанных функций и решение заданных задач, оценить их сложность, используя О-символику.

Вариант 8

1. Вводя в цикле по 5 оценок каждого студента, получить число студентов, которые не имеют оценок 2 и 3. В группе учится n студентов.

namespace Labarator\_1

{

public class Student : INotifyPropertyChanged

{

#region Переменные

/// <summary>

/// Имя студента

/// </summary>

private string name;

/// <summary>

/// Фамилия студента

/// </summary>

private string surname;

/// <summary>

/// Оценка по предмету ПВС

/// </summary>

private int pvsValue;

/// <summary>

/// Оценка по предмету ТРАРИЗ

/// </summary>

private int trarizValue;

/// <summary>

/// Оценка по предмету СП

/// </summary>

private int spValue;

/// <summary>

/// Оценка по предмету ОЕМ

/// </summary>

private int oemValue;

/// <summary>

/// Оценка по предмету ИПЗ

/// </summary>

private int ipzValue;

#endregion

#region Свойства

/// <summary>

/// Имя студента

/// </summary>

public string Name

{

get { return name; }

set

{

name = value;

OnPropertyChanged("Name");

}

}

/// <summary>

/// Фамилия студента

/// </summary>

private string Surname

{

get { return surname; }

set

{

surname = value;

OnPropertyChanged("Surname");

}

}

/// <summary>

/// Оценка по предмету ПВС

/// </summary>

private int PvsValue

{

get { return pvsValue; }

set

{

pvsValue = value;

OnPropertyChanged("PvsValue");

}

}

/// <summary>

/// Оценка по предмету ТРАРИЗ

/// </summary>

private int TrarizValue

{

get { return trarizValue; }

set

{

trarizValue = value;

OnPropertyChanged("TrarizValue");

}

}

/// <summary>

/// Оценка по предмету СП

/// </summary>

private int SpValue

{

get { return spValue; }

set

{

spValue = value;

OnPropertyChanged("SpValue");

}

}

/// <summary>

/// Оценка по предмету ОЕМ

/// </summary>

private int OemValue

{

get { return oemValue; }

set

{

oemValue = value;

OnPropertyChanged("OemValue");

}

}

/// <summary>

/// Оценка по предмету ИПЗ

/// </summary>

private int IpzValue

{

get { return ipzValue; }

set

{

ipzValue = value;

OnPropertyChanged("IpzValue");

}

}

/// <summary>

///

/// </summary>

public event PropertyChangedEventHandler PropertyChanged;

/// <summary>

///

/// </summary>

/// <param name="propertyName"></param>

protected virtual void OnPropertyChanged(string propertyName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

#endregion

#region Конструктор

/// <summary>

///

/// </summary>

/// <param name="n"></param>

/// <param name="s"></param>

/// <param name="pvs"></param>

/// <param name="ipz"></param>

/// <param name="oem"></param>

/// <param name="trariz"></param>

/// <param name="sp"></param>

public Student(string n, string s, int pvs, int ipz, int oem, int trariz, int sp)

{

Name = name;

Surname = s;

PvsValue = pvs;

IpzValue = ipz;

OemValue = oem;

SpValue = sp;

TrarizValue = trariz;

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Forms;

using System.Xml.Linq;

using System.Xml.Serialization;

namespace Labarator\_1\_1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

studentList = new List<Student>();

txtStudentListCount.Text = Convert.ToString(studentList.Count);

}

List<Student> studentList;

private int counter = 0;

#region Методы

private bool CompareStudentValue(Student student, int min)

{

if (

student.IpzValue <= min ||

student.SpValue <= min ||

student.TrarizValue <= min ||

student.OetValue <= min ||

student.PvsValue <= min

)

return true;

return false;

}

private void ClearFields()

{

txtName.Clear();

txtSurname.Clear();

txtPvs.Clear();

txtIpz.Clear();

txtSp.Clear();

txtTrariz.Clear();

txtOet.Clear();

}

#endregion

#region Кнопки

/// <summary>

/// Кнопка сохранения значения

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnSave\_Click(object sender, System.EventArgs e)

{

try

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "XML-File | \*.xml";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

XElement xmlList = new XElement("Root",

from student in studentList

select

new XElement("Student",

new XElement("Name", student.Name),

new XElement("Surname", student.Surname),

new XElement("IpzValue", student.IpzValue),

new XElement("SpValue", student.SpValue),

new XElement("PvsValue", student.PvsValue),

new XElement("TrarizValue", student.TrarizValue),

new XElement("OetValue", student.OetValue)

));

xmlList.Save(saveFileDialog.FileName);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Кнопка очистки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnClean\_Click(object sender, System.EventArgs e)

{

ClearFields();

}

/// <summary>

/// Поиск

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnFind\_Click(object sender, System.EventArgs e)

{

counter = 0;

if (string.IsNullOrEmpty(txtMinValue.Text))

MessageBox.Show("Введите оценку ");

else

{

foreach (Student student in studentList)

{

if (CompareStudentValue(student, int.Parse(txtMinValue.Text)))

counter++;

}

txtResult.Text = Convert.ToString(counter);

}

}

/// <summary>

/// Обработчик закрытия окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnCancel\_Click(object sender, System.EventArgs e)

{

Close();

}

/// <summary>

/// Добавление студента в список

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnAdd\_Click(object sender, EventArgs e)

{

if (

string.IsNullOrEmpty(txtName.Text) ||

string.IsNullOrEmpty(txtSurname.Text) ||

string.IsNullOrEmpty(txtPvs.Text) ||

string.IsNullOrEmpty(txtIpz.Text) ||

string.IsNullOrEmpty(txtSp.Text) ||

string.IsNullOrEmpty(txtTrariz.Text) ||

string.IsNullOrEmpty(txtOet.Text)

)

MessageBox.Show("Введите значения");

else

{

try

{

studentList.Add(new Student(

txtName.Text,

txtSurname.Text,

int.Parse(txtPvs.Text),

int.Parse(txtIpz.Text),

int.Parse(txtTrariz.Text),

int.Parse(txtSp.Text),

int.Parse(txtOet.Text)

));

txtStudentListCount.Text = Convert.ToString(studentList.Count);

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

#endregion

#region Обработчики нажатия

/// <summary>

/// Обработчик вводимых данных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void txtPvs\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((!Char.IsDigit(number) || (number < 49) && (number > 53)))

{

e.Handled = true;

}

}

/// <summary>

/// Обработчик вводимых данных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void txtIpz\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((!Char.IsDigit(number) || (number < 49 || number > 53)))

{

e.Handled = true;

}

}

/// <summary>

/// Обработчик вводимых данных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void txtSp\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((!Char.IsDigit(number) || (number < 49 || number > 53)))

{

e.Handled = true;

}

}

/// <summary>

/// Обработчик вводимых данных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void txtOet\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((!Char.IsDigit(number) || (number < 49 || number > 53)))

{

e.Handled = true;

}

}

/// <summary>

/// Обработчик вводимых данных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void txtTrariz\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((!Char.IsDigit(number) || (number < 49 || number > 53)))

{

e.Handled = true;

}

}

/// <summary>

/// Обработчик вводимых данных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void txtLowLevel\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((!Char.IsDigit(number) || (number < 49 || number > 53)))

{

e.Handled = true;

}

}

#endregion

private void btnLoadXml\_Click(object sender, EventArgs e)

{

try

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "xml|\*.xml|all files|\*.\*";

DialogResult res = openFileDialog.ShowDialog();

if (res == DialogResult.OK)

{

XDocument xdoc = XDocument.Load(openFileDialog.FileName);

studentList = (from lv1 in xdoc.Descendants("Student")

select new Student

{

Name = lv1.Element("Name").Value,

Surname = lv1.Element("Surname").Value,

PvsValue = int.Parse(lv1.Element("PvsValue").Value),

IpzValue = int.Parse(lv1.Element("IpzValue").Value),

OetValue = int.Parse(lv1.Element("OetValue").Value),

TrarizValue = int.Parse(lv1.Element("TrarizValue").Value),

SpValue = int.Parse(lv1.Element("SpValue").Value)

}).ToList();

txtStudentListCount.Text = Convert.ToString(studentList.Count);

}

}

catch

{

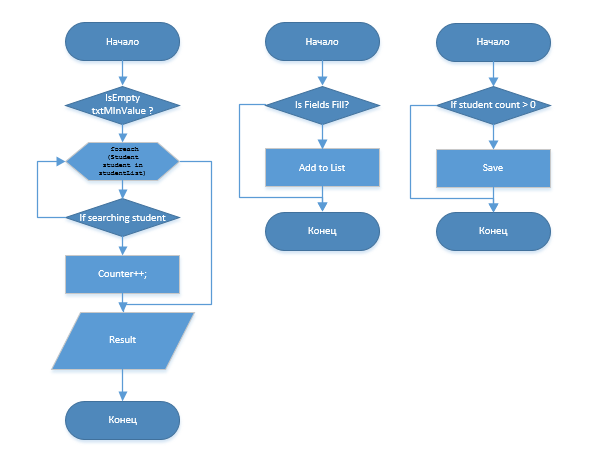
MessageBox.Show("invalid input");

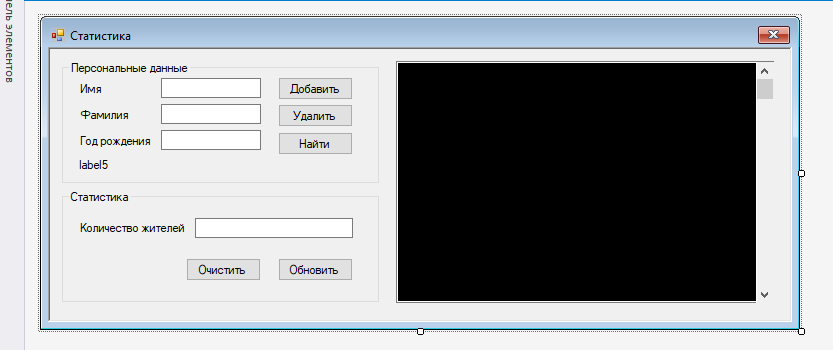
}

}

}

}





1. Результаты переписи населения хранятся в памяти ЭВМ. Используя массивы фамилий и года рождения, напечатать фамилии и подсчитать общее число жителей, родившихся раньше 1930 года

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Labarator\_1\_2

{

public class Liver

{

/// <summary>

/// Имя

/// </summary>

public string Name { get; set; }

/// <summary>

/// Фамилия

/// </summary>

public string Surname { get; set; }

/// <summary>

/// Год рождения

/// </summary>

public int BDay { get; set; }

/// <summary>

/// Конструктор

/// </summary>

/// <param name="n">Имя</param>

/// <param name="s">Фамилия</param>

/// <param name="a">Год рождения</param>

public Liver(string n, string s,int a)

{

Name = n;

Surname = s;

BDay = a;

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Labarator\_1\_2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

LiverList = new List<Liver>();

}

/// <summary>

/// Список жителей

/// </summary>

private List<Liver> LiverList { get; set; }

/// <summary>

/// Проверка введены ли данные

/// </summary>

/// <returns></returns>

private bool IsFieldsFill()

{

if (string.IsNullOrEmpty(txtbBDay.Text) ||

string.IsNullOrEmpty(txtName.Text) ||

string.IsNullOrEmpty(txtSurname.Text))

{

MessageBox.Show("Не все поля заполнены");

return false;

}

else

return true;

}

/// <summary>

/// Удаление данных

/// </summary>

private void ClearData()

{

txtbBDay.Clear();

txtName.Clear();

txtSurname.Clear();

}

/// <summary>

/// Поиск в списке

/// </summary>

/// <param name="Name"></param>

/// <param name="Surname"></param>

private Liver FindLiver(string Name, string Surname)

{

foreach (Liver liver in LiverList)

{

if (liver.Name == Name && liver.Surname == Surname)

return liver;

}

return null;

}

/// <summary>

/// Кнопка добавления нового жителя

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnAdd\_Click(object sender, EventArgs e)

{

lblFindResult.Visible = false;

if (IsFieldsFill())

LiverList.Add(new Liver(txtName.Text, txtSurname.Text, int.Parse(txtbBDay.Text)));

ClearData();

}

/// <summary>

/// Кнопка удаления жителя

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnRemove\_Click(object sender, EventArgs e)

{

lblFindResult.Visible = false;

if (IsFieldsFill())

LiverList.Remove(FindLiver(txtName.Text, txtSurname.Text));

ClearData();

}

/// <summary>

/// Поиск жителя

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnFind\_Click(object sender, EventArgs e)

{

lblFindResult.Visible = false;

if (FindLiver(txtName.Text, txtSurname.Text) == null)

lblFindResult.Text = "Такого жителя нет";

else

{

int index = LiverList.IndexOf(FindLiver(txtName.Text, txtSurname.Text));

lblFindResult.Text = "Житель под индексом {index}";

}

lblFindResult.Visible = true;

}

/// <summary>

/// Пересчет списка

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnUpdate\_Click(object sender, EventArgs e)

{

lblFindResult.Visible = false;

txtConsole.Clear();

txtResult.Text = Convert.ToString(DisplayList());

}

private int DisplayList()

{

string format = "{0,5}{1,15}{2,30}{3,35}";

txtConsole.Text += String.Format(format, "#", "Name", "Surname", "BirthDay");

int counter = 0;

foreach(Liver liver in LiverList)

{

counter++;

txtConsole.Text += Environment.NewLine;

txtConsole.Text += String.Format(format, counter, liver.Name, liver.Surname, liver.BDay);

}

return counter;

}

/// <summary>

/// Очистить консоль и поля

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnClear\_Click(object sender, EventArgs e)

{

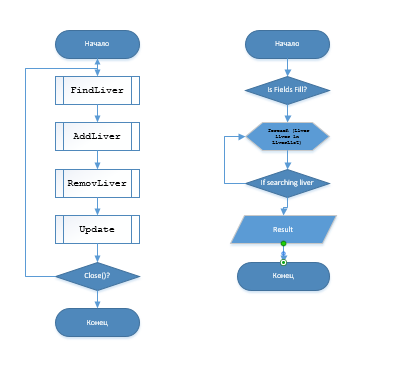
ClearData();

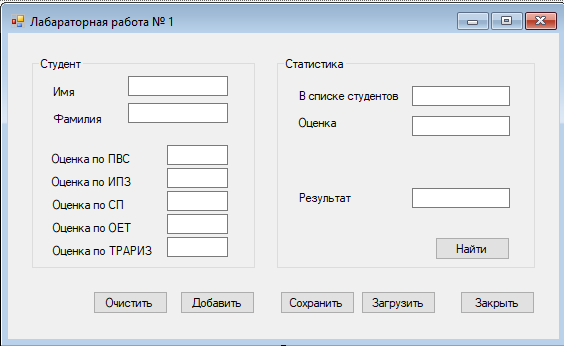
txtConsole.Clear();

}

}

}





Выводы:

В ходе выполнения лабораторной работы № 1 овладел навыками определения времени и порядка степени роста алгоритма, приобрел навыки в разработке блок-схем, псевдокода алгоритмов и написании программ с использованием ветвлений и циклов.