**МИНИСТЕРСТВО ТРАНСПОРТА**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ**

**АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО**

**ОБРАЗОВАНИЯ**

**“РОССИЙСКИЙ УНИВЕРСИТЕТ**

**ТРАНСПОРТА” (МИИТ)**

**Институт транспортной техники и систем**

**управления**

**(ИТТСУ)**

**Кафедра “Управление и защита информации”**

**Тема: “Лабораторная работа №4”**

**Выполнили студенты группы ТКИ-342**

**Савин Лев Андреевич   
Потапов Артём Константинович**

**Специальность 10.05.01 “Компьютерная безопасность”**

**Преподаватель: к.т.н. Сафронов А.И.**

**Москва**

**2024**

1. Формулировка цели работы:

Разработать программное обеспечение, которое по выбранной дате в календаре определит номер соответствующей недели в учебном семестре с учетом опорных дат 1 сентября и 7 февраля.

2. Описание задачи:

Необходимо создать программу, которая будет принимать на вход выбранную дату и на основе опорных дат 1 сентября и 7 февраля определит номер недели в учебном семестре. Если опорная дата выпала на выходные дни, то первой неделей считается ближайший будний день предстоящей недели.

3. Составление диаграммы классов:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

4. Составление сети Петри запрограммированного технологического процесса:   
Изображение выглядит как диаграмма, линия, снимок экрана, Шрифт

Автоматически созданное описание  
5. Составление схем алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (метка).  
Изображение выглядит как диаграмма, круг, снимок экрана, дизайн

Автоматически созданное описание

6. Подбор тестовых примеров:

* Для опорных дат: 1 сентября, 7 февраля
* Для случаев, когда опорная дата выпала на выходные дни: суббота или воскресенье
* Для произвольных дат внутри учебного семестра

7. Листинг (код) программного обеспечения:

public partial class FrmMain : Form

{

public FrmMain()

{

InitializeComponent();

}

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

{

WindowState = FormWindowState.Minimized;

}

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

{

\_ = WindowState == FormWindowState.Normal ? WindowState = FormWindowState.Maximized : WindowState = FormWindowState.Normal;

\_ = btnMaximize.Text == "Maximize" ? btnMaximize.Text = "Normalize" : btnMaximize.Text = "Maximize";

}

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

{

Application.Exit();

}

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

{

StreamWriter streamWriter = new StreamWriter("dates.txt", true);

if (mcActive.SelectionStart.ToShortDateString() == mcActive.SelectionEnd.ToShortDateString())

{

streamWriter.WriteLine(mcActive.SelectionStart.ToShortDateString());

}

else

{

DateTime holder = mcActive.SelectionStart;

while (mcActive.SelectionStart.ToShortDateString() != mcActive.SelectionEnd.ToShortDateString())

{

streamWriter.WriteLine(mcActive.SelectionStart.ToShortDateString());

mcActive.SelectionStart = mcActive.SelectionStart.AddDays(1);

}

streamWriter.WriteLine(mcActive.SelectionStart.ToShortDateString());

mcActive.SelectionStart = holder;

}

streamWriter.Close();

}

private void FrmMain\_Load(object sender, EventArgs e)

{

using (StreamReader streamReader = new StreamReader("ttMcActive.txt"))

{

ttBase.SetToolTip(mcActive, streamReader.ReadToEnd());

}

using (StreamReader streamReader = new StreamReader("ttBtnFileLoad.txt"))

{

ttBase.SetToolTip(btnFileLoad, streamReader.ReadToEnd());

}

using (StreamReader streamReader = new StreamReader("ttBtnMinimize.txt"))

{

ttBase.SetToolTip(btnMinimize, streamReader.ReadToEnd());

}

using (StreamReader streamReader = new StreamReader("ttBtnMaximize.txt"))

{

ttBase.SetToolTip(btnMaximize, streamReader.ReadToEnd());

}

using (StreamReader streamReader = new StreamReader("ttBtnClose.txt"))

{

ttBase.SetToolTip(btnClose, streamReader.ReadToEnd());

}

mcActive.SelectionStart = mcActive.SelectionStart.AddDays(-1);

mcActive.SelectionStart = mcActive.SelectionStart.AddDays(1);

}  
 private void mcActive\_DateChanged(object sender, DateRangeEventArgs e)

{

if (mcActive.SelectionStart.ToShortDateString() != mcActive.SelectionEnd.ToShortDateString())

{

tbWeek.Text = "You must select only one day!";

}

else

{

int month = Convert.ToInt32(mcActive.SelectionStart.ToShortDateString().Remove(5).Remove(0, 3));

int day = Convert.ToInt32(mcActive.SelectionStart.ToShortDateString().Remove(2));

int year = Convert.ToInt32(mcActive.SelectionStart.ToShortDateString().Remove(0, 6));

int flag = -1;

if (month > 8)

{

flag = 1;

}

else if (month > 2 && month < 7 || month == 2 && day > 6)

{

flag = 0;

}

else if (flag != 0 && flag != 1)

{

tbWeek.Text = "It's a holiday time! Let's party!";

return;

}

switch (CalculateWeekCount(year, Convert.ToBoolean(flag)))

{

case 1:

tbWeek.Text = "In that day it is a II week.";

break;

case 0:

tbWeek.Text = "In that day it is a I week.";

break;

case -1:

tbWeek.Text = "The last days of holidays!!!";

break;

}

}

}

private int CalculateWeekCount(int year, bool firstSemester)

{

//semester: false -- first, true -- second

int count = 0;

DateTime etalon = new DateTime(year, 2, 7);

if (firstSemester)

{

DateTime change = new DateTime(year, 9, 1);

etalon = etalon.AddDays(Convert.ToInt32((change - etalon).TotalDays));

}

if (etalon.DayOfWeek == DayOfWeek.Sunday)

{

etalon = etalon.AddDays(1);

}

else if (etalon.DayOfWeek == DayOfWeek.Saturday)

{

etalon = etalon.AddDays(2);

}

else

{

while (etalon.DayOfWeek != DayOfWeek.Monday)

{

etalon = etalon.AddDays(-1);

count++; //count number of working days in first week

}

etalon = etalon.AddDays(count);

}

int countOfWeeks = Convert.ToInt32((mcActive.SelectionStart - etalon).TotalDays) + count;

if (countOfWeeks >= 0)

{

countOfWeeks /= 7;

countOfWeeks %= 2;

}

else

{

countOfWeeks = -1;

}

return countOfWeeks;

}

private void tbWeek\_MouseEnter(object sender, EventArgs e)

{

using (StreamReader streamReader = new StreamReader("ttTbWeek.txt"))

{

ttBase.SetToolTip(tbWeek, streamReader.ReadToEnd());

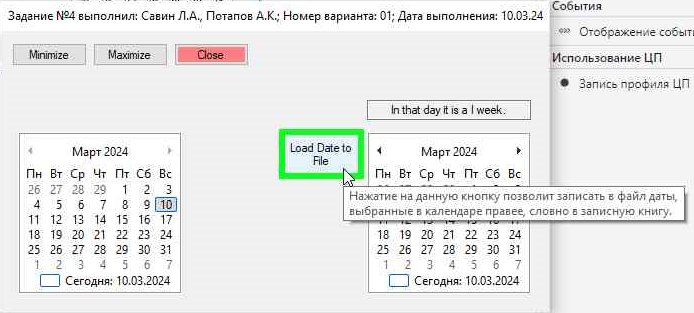
}

}

}

8. Графический пользовательский интерфейс:

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание  
Меню скомпоновано в графической среде  
  
У каждого активного элемента есть окно подсказки

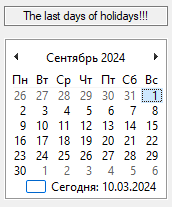
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание  
Файл с записанными датами

9. Расчёт тестовых примеров:  
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание  
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание  
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание  


10. Вывод:

Разработанное программное обеспечение успешно выполняет задачу определения номера недели в учебном семестре с учетом опорных дат и правил, установленных в условии. Оно позволяет пользователю удобно выбирать даты и получать соответствующие номера недель.