МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**Национальный исследовательский ядерный университет «МИФИ»**

|  |  |
| --- | --- |
|  | |
|  | **Институт  интеллектуальных кибернетических систем** |
| **Кафедра №22 «Кибернетика»** |
| Направление подготовки 09.03.04 Программная инженерия | |
| **Отчет по курсу:** | |
| **«Технологии промышленной разработки программного обеспечения. Основные понятия»** | |

Лабораторная работа 3

|  |  |  |
| --- | --- | --- |
| Группа | M24-514 | |
| Студент | Фонов Н.Д. | |  |
|  | (ФИО) | |
| Руководитель | Гагарин А.П. | |  |
|  | (ФИО) | |
|  |  | |  |
|  | |
|  | |
|  | |
| **Москва 2024** | |

**1. Описание.**

Лабораторная работа 3 представлена в виде C# проекта (windows forms). Была выполнена доработка для обеспечения возможности перебора элементов. Перебор осуществляется последовательным всплыванием окон с информацией о том, выбран данный элемент или нет.

**2. Структура проекта.**

* lab3
  + Form1.cs
  + Program.cs
  + MiniList.cs
* Прочие файлы.

В файле Form1.cs описан класс формы для перебора элементов, его методы, вся бизнес-логика.

В файле Program.cs описан класс Program, который отвечает за создание формы.

В файле MiniList.cs вынесен одноименный класс перебираемого списка.

**3. Архитектурные решения.**

Поскольку в проекте требовалось реализовать класс с реализованным функционалом перебора, то данный класс был вынесен в отдельный файл. Для реализации перебора данный класс реализует интерфейс IEnumerable. Соответственно, данный класс реализует требуемые для перебора методы.

**4. Тест-рейсы.**

1. Используя пустой список. Ожидается, что результатом будет сообщение, что список перебран.

2. Использование непустого списка без выбранных элементов. Ожидается, что на каждый элемент будет выведено сообщение о том, что элемент не выбран.

3. Использование непустого списка с несколькими (>0) выбранными элементами. Ожидается, что на каждый не выбранный элемент будет выведено сообщение о том, что элемент не выбран. На каждый выбранный элемент будет выведено сообщение о том, что элемент выбран.

4. Использование непустого списка только с выбранными элементами. Ожидается, что на каждый элемент будет выведено сообщение о том, что элемент выбран.

**5. Тестирование и отладка.**

Для проверки корректности работы программы применялась ручная отладка по написанным выше тест-кейсам.

Результат ручной отладки показал, что программа работает корректно на всех заявленных выше вариантах использования шаблона.

**6. Реализация.**

На рисунке 1 представлена работа кнопки меню создания списка.

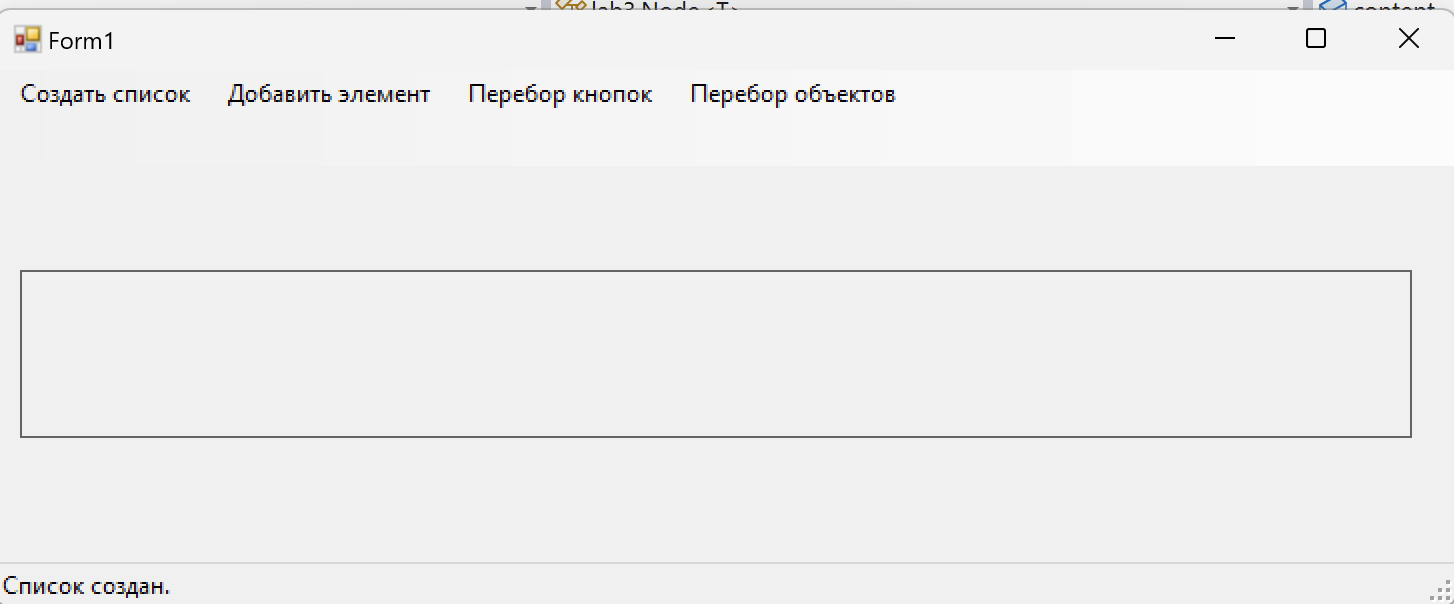


Рис. 1 – Создание списка

На рисунке 2 представлено создание элемента на уже созданном списке.

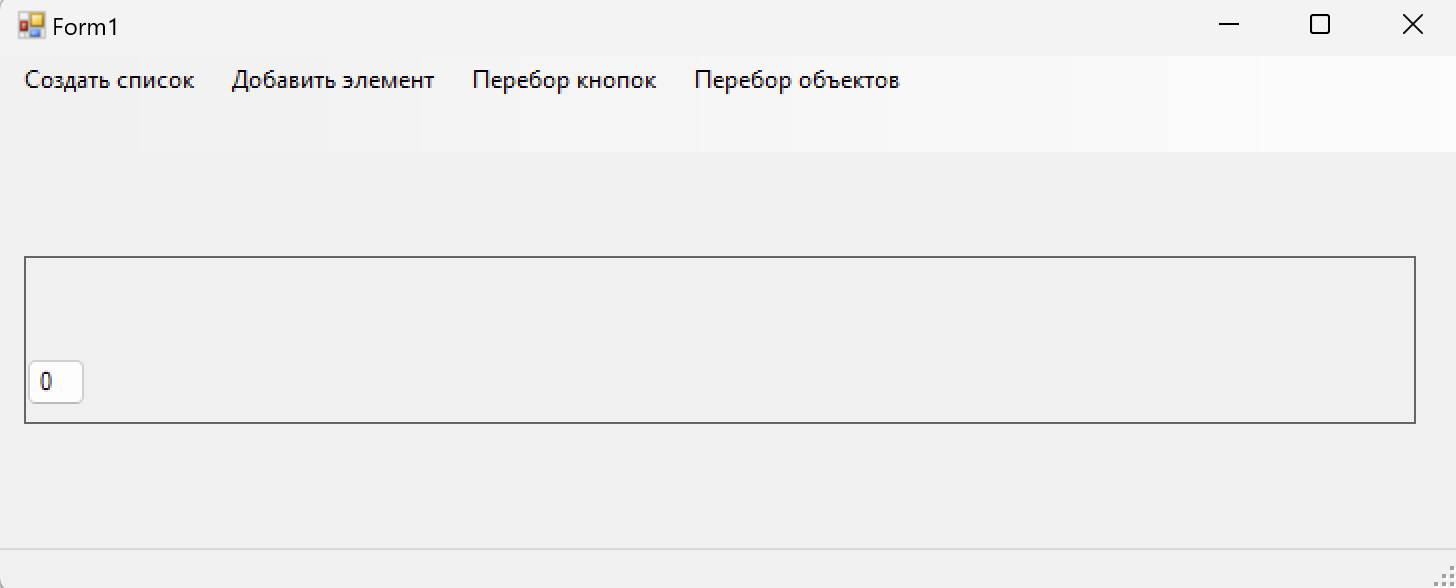


Рис. 2 – Создание элемента

На рисунке 3 представлен выбор элементов 40, 80 и 120.

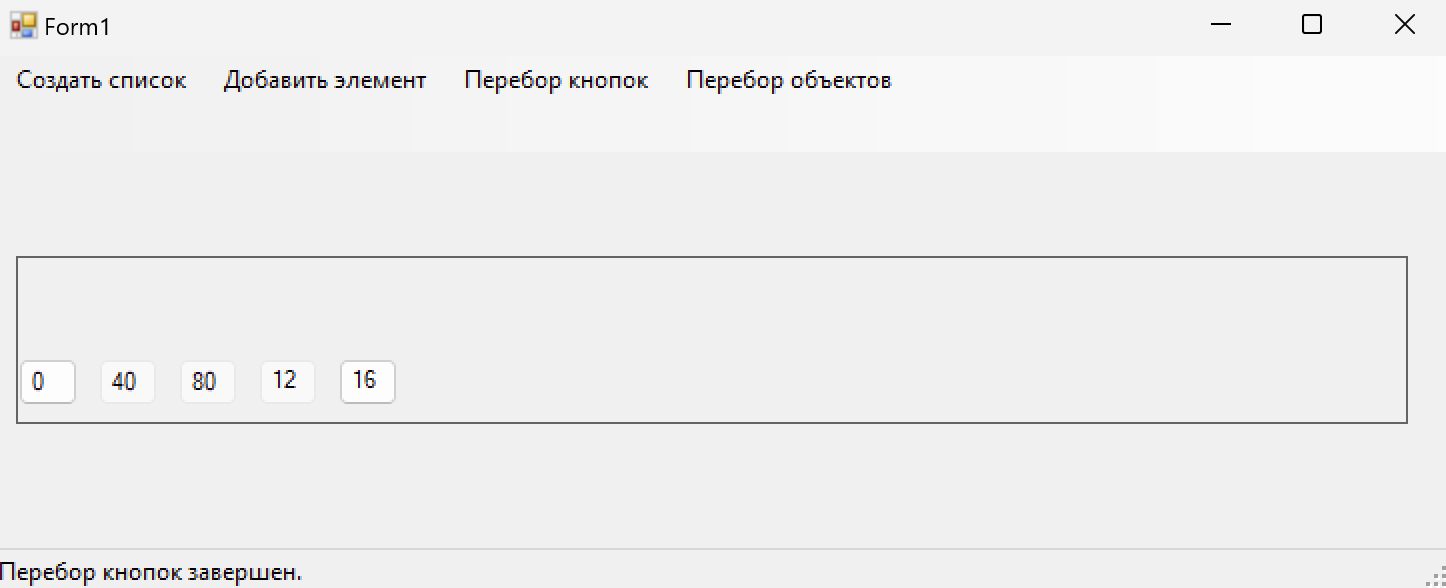


Рис. 3 – выбор элементов (отмечаются серым)

На рисунках 4-6 представлен пример перебора чек-боксов списка (как отмеченных, так и не отмеченных).

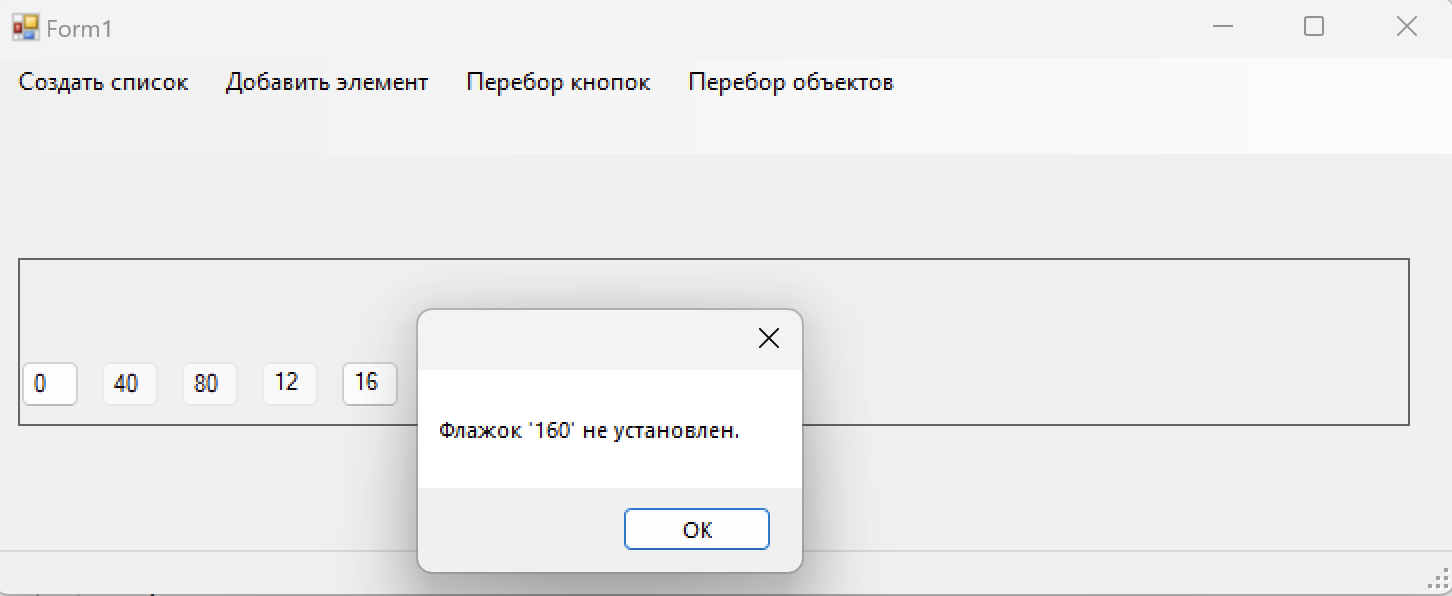


Рис. 4 – Перебор элементов 1

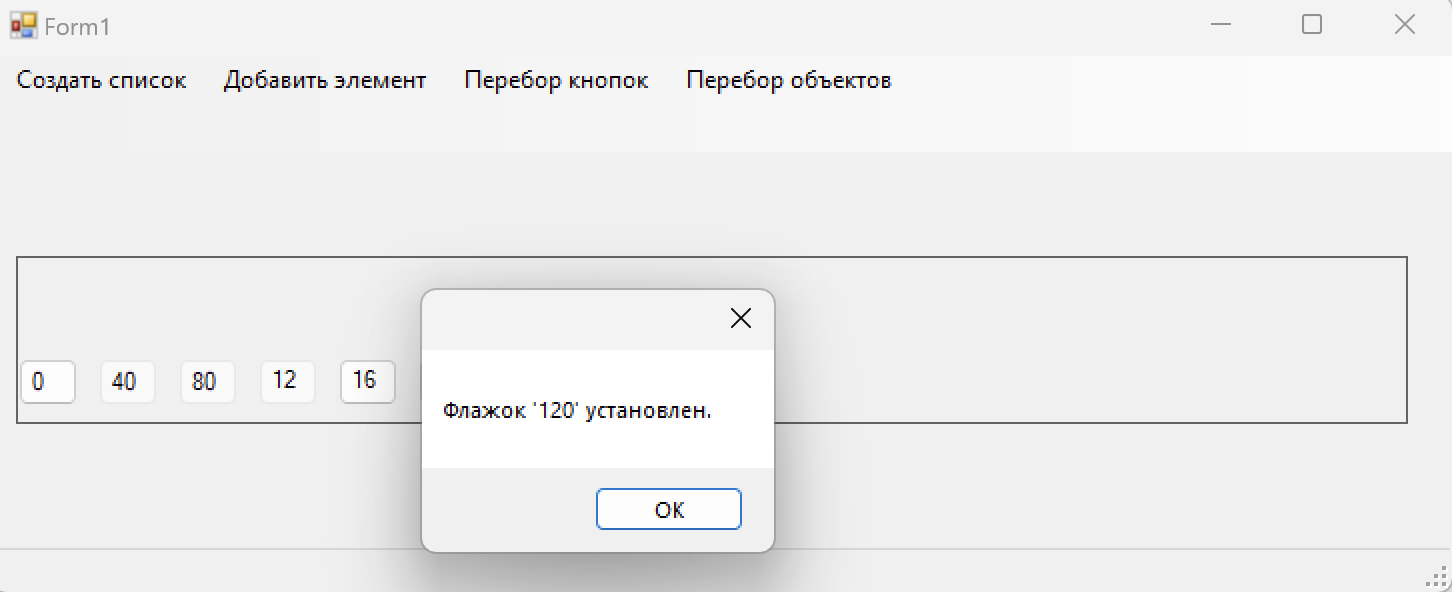


Рис. 5 – Перебор элементов 2

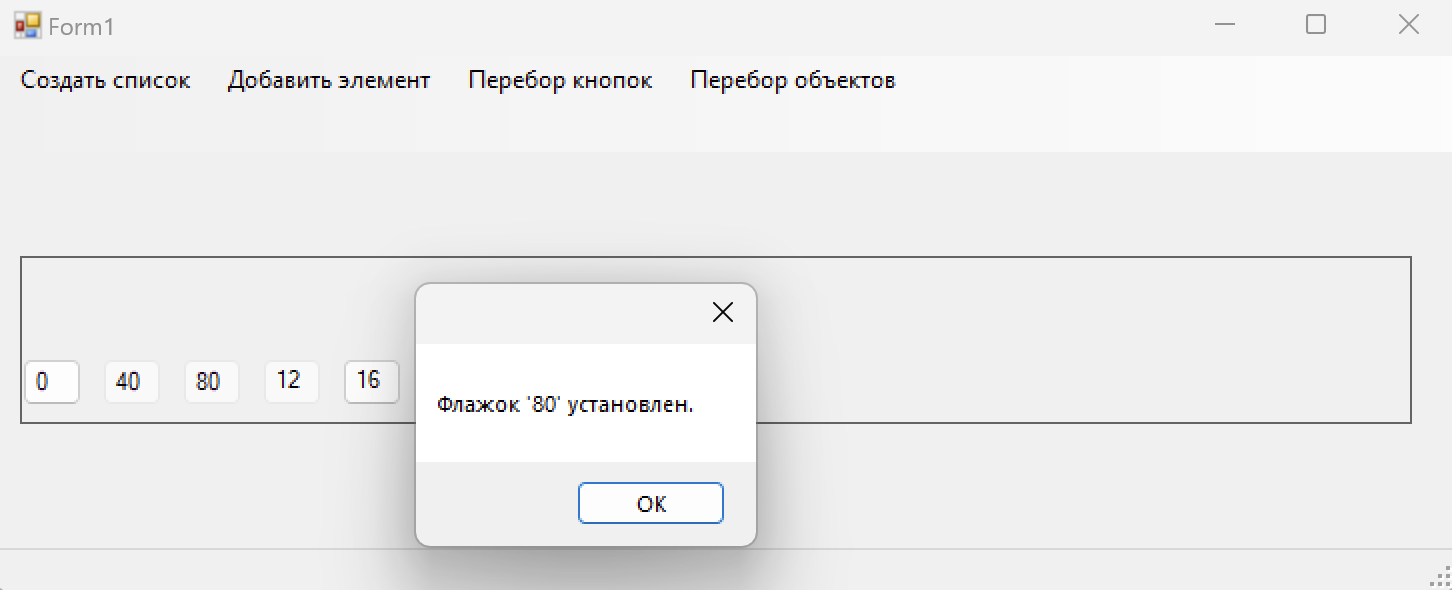


Рис. 6 – Перебор элементов 3

**Приложение 1. Программный код MiniList.**

using System;

using System.Collections;

using System.Collections.Generic;

namespace lab3

{

public class Node<T>

{

public T content;

public Node<T> Next = null;

}

internal class MiniList<T> : IEnumerable<T>

{

static int num = 0;

public Node<T> Top;

public int Append(T s)

{

Node<T> p = new Node<T>();

p.content = s;

if (Top != null) p.Next = Top;

Top = p;

return num++;

}

public IEnumerator<T> GetEnumerator() => new MiniListEnum<T>(Top);

IEnumerator IEnumerable.GetEnumerator() => new MiniListEnum<T>(Top);

}

internal class MiniListEnum<T> : IEnumerator<T>

{

public Node<T> Top;

public Node<T> ENode;

public MiniListEnum(Node<T> top)

{

Node<T> p = new Node<T>();

p.Next = top;

Top = p;

ENode = p;

}

public T Current

{

get

{

return ENode.content;

}

}

object IEnumerator.Current => ENode.content;

public void Dispose()

{

GC.SuppressFinalize(this);

}

public bool MoveNext()

{

ENode = ENode.Next;

return ENode != null;

}

public void Reset()

{

ENode = Top;

}

}

}

**Приложение 2. Программный код Form1.**

using System;

using System.Windows.Forms;

namespace lab3

{

public partial class Form1 : Form

{

MiniList<CheckBox> v;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e) { }

private void toolStripMenuItem1\_Click(object sender, EventArgs e)

{

v = new MiniList<CheckBox>();

toolStripStatusLabel1.Text = "Список создан.";

}

private void toolStripMenuItem2\_Click(object sender, EventArgs e)

{

if (v != null)

{

CheckBox vcb = new CheckBox();

vcb.Appearance = Appearance.Button;

vcb.FlatStyle = FlatStyle.Standard;

vcb.Top = 50;

vcb.Width = 30;

vcb.Left = 40 \* v.Append(vcb);

vcb.Text = Convert.ToString(vcb.Left);

this.panel1.Controls.Add(vcb);

toolStripStatusLabel1.Text = "";

}

else toolStripStatusLabel1.Text = "Список не создан.";

}

private void toolStripMenuItem3\_Click(object sender, EventArgs e)

{

{

if (v == null)

{

toolStripStatusLabel1.Text = "Список пуст.";

return;

}

foreach (CheckBox checkBox in v)

{

if (checkBox.Checked)

{

MessageBox.Show($"Флажок '{checkBox.Text}' установлен.");

}

else

{

MessageBox.Show($"Флажок '{checkBox.Text}' не установлен.");

}

}

toolStripStatusLabel1.Text = "Перебор кнопок завершен.";

}

}

private void toolStripMenuItem4\_Click(object sender, EventArgs e){ }

}

}