МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Гомельский государственный технический

университет имени П.О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Отчёт по лабораторной работе №6

По дисциплине «Объектно-ориентированное программирование»

**«****Коллекции. Обобщенные типы. Итераторы»**

Выполнил:

Студент группы ИТП-21

Коркуц С. И.

Принял:

ассистент Карабчикова Е.А.

Гомель 2020

**Цель работы:** изучить основы коллекция, обобщенных типов, итераторы.

**Задание:**

Разработать коллекцию. Обязательное наличие итератора. Также необходимо реализовать интерфейс ICollection. Предусмотреть возможность хранения данных любого типа (классы и структуры), поддерживающего возможные сравнения. Все коллекции должны быть реализованы без использования стандартных коллекций.

Разработать UI-интерфейс WinForms (четные варианты) или WPF-приложение (нечетные варианты) и продемонстрировать работу коллекции на простых типах и на разработанном типе из лабораторной работы 2.



Рисунок 1 – Вариант задания

На рисунке 2 изображена структура решения.

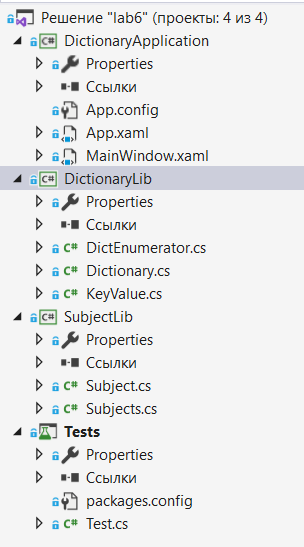


Рисунок 2 – Структура решения

При запуске программы открывается основное меню (Рисунок 3).

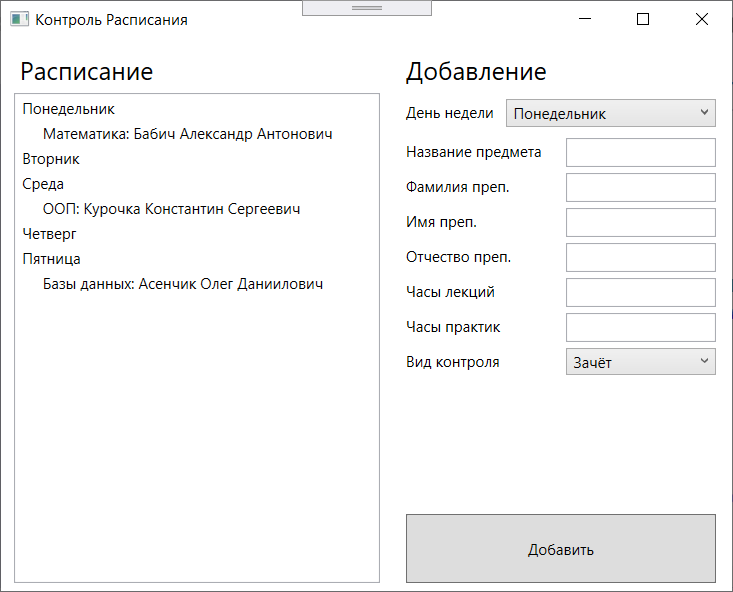


Рисунок 3 – Основное меню программы

Добавление предмета в расписание на рисунке 4.

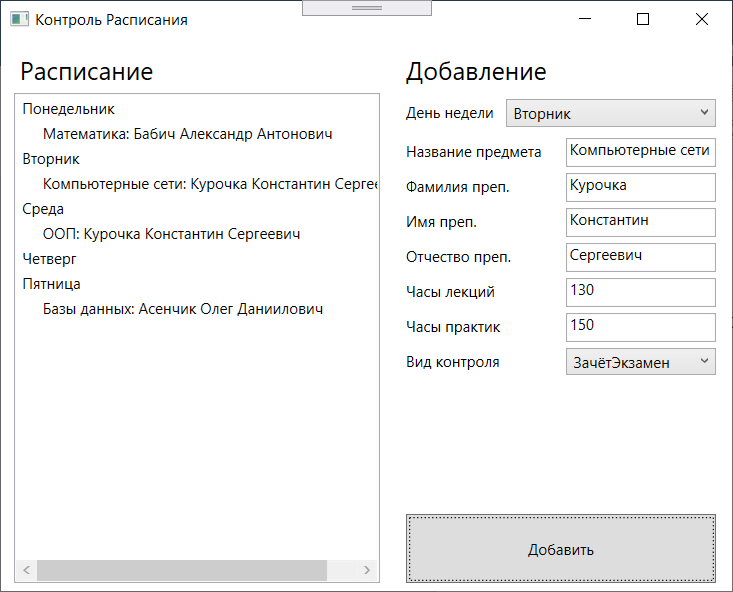


Рисунок 4 – расписание после добавления предмета

**Вывод:** изучены основы синтаксиса объектно-ориентированного языка программирования, основы коллекция, обобщенных типов, итераторы.

**ПРИЛОЖЕНИЕ А**

Листинг класса *DictEnumerator*.

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DictionaryLib

{

class DictEnumerator<TKey, TValue> : IEnumerator<KeyValue<TKey, TValue>>

{

KeyValue<TKey, TValue>[] keysValues;

int position = -1;

public DictEnumerator(KeyValue<TKey, TValue>[] keysValues)

{

this.keysValues = keysValues;

}

public KeyValue<TKey, TValue> Current

{

get

{

if (position == -1 || position > keysValues.Length)

throw new InvalidOperationException();

return keysValues[position];

}

}

object IEnumerator.Current => throw new NotImplementedException();

public bool MoveNext()

{

if (position < keysValues.Length - 1)

{

position += 1;

return true;

}

else

return false;

}

public void Reset()

{

position = -1;

}

public void Dispose() { }

}

}

**ПРИЛОЖЕНИЕ Б**

Листинг класса *Dictionary*.

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DictionaryLib

{

public class Dictionary<TKey, TValue> : IEnumerable<KeyValue<TKey, TValue>>,

ICollection<KeyValue<TKey, TValue>>

{

KeyValue<TKey, TValue>[] keysValues;

public Dictionary(KeyValue<TKey, TValue>[] keysValues)

{

this.keysValues = keysValues;

}

public Dictionary()

{

this.keysValues = new KeyValue<TKey, TValue>[0];

}

public IEnumerator<KeyValue<TKey, TValue>> GetEnumerator()

{

return new DictEnumerator<TKey, TValue>(keysValues);

}

IEnumerator IEnumerable.GetEnumerator()

{

throw new NotImplementedException();

}

public int Count => keysValues.Length;

public bool IsReadOnly => false;

public void Add(KeyValue<TKey, TValue> item)

{

KeyValue<TKey, TValue>[] newArr = new KeyValue<TKey, TValue>[this.Count() + 1];

for(int i = 0; i < this.Count; i++)

{

newArr[i] = keysValues[i];

}

newArr[this.Count()] = item;

keysValues = newArr;

}

public void Add(TKey key, TValue value)

{

KeyValue<TKey, TValue>[] newArr = new KeyValue<TKey, TValue>[this.Count() + 1];

for (int i = 0; i < this.Count; i++)

{

newArr[i] = keysValues[i];

}

newArr[this.Count()] = new KeyValue<TKey, TValue>(key, value);

keysValues = newArr;

}

public void Clear()

{

keysValues = new KeyValue<TKey, TValue>[0];

}

public bool Contains(KeyValue<TKey, TValue> item)

{

for(int i = 0; i < this.Count(); i++)

{

if(keysValues[i] == item) //Под вопросом будет ли оно правильно сравнивать

{

return true;

}

}

return false;

}

public bool Contains(TKey key)

{

for (int i = 0; i < this.Count(); i++)

{

if (keysValues[i].Key.Equals(key)) //Под вопросом будет ли оно правильно сравнивать

{

return true;

}

}

return false;

}

public void CopyTo(KeyValue<TKey, TValue>[] array, int arrayIndex)

{

for(int i = arrayIndex; i < this.Count && i - arrayIndex < array.Length; i++)

{

array[i - arrayIndex] = keysValues[i];

}

}

public bool Remove(KeyValue<TKey, TValue> item)

{

if(this.Contains(item))

{

KeyValue<TKey, TValue>[] newArr = new KeyValue<TKey, TValue>[this.Count() - 1];

int flag = 0;

for (int i = 0; i < this.Count; i++)

{

if(keysValues[i] == item)

{

flag = 1;

continue;

}

newArr[i - flag] = keysValues[i];

}

keysValues = newArr;

return true;

}

return false;

}

public bool Remove(TKey key)

{

if (this.GetKeys().Contains(key))

{

KeyValue<TKey, TValue>[] newArr = new KeyValue<TKey, TValue>[this.Count() - 1];

int flag = 0;

for (int i = 0; i < this.Count; i++)

{

if (keysValues[i].Key.Equals(key))

{

flag = 1;

continue;

}

newArr[i - flag] = keysValues[i];

}

keysValues = newArr;

return true;

}

return false;

}

public TKey[] GetKeys()

{

TKey[] keys = new TKey[this.Count];

for(int i = 0; i < this.Count; i++)

{

keys[i] = keysValues[i].Key;

}

return keys;

}

public TValue[] GetValues()

{

TValue[] values = new TValue[this.Count];

for (int i = 0; i < this.Count; i++)

{

values[i] = keysValues[i].Value;

}

return values;

}

public TValue this[TKey key]

{

get

{

foreach(KeyValue<TKey, TValue> keyValue in keysValues)

{

if(keyValue.Key.Equals(key))

{

return keyValue.Value;

}

}

throw new Exception("Нет такого ключа");

}

}

}

}

**ПРИЛОЖЕНИЕ В**

Листинг класса *KeyValue*.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DictionaryLib

{

public class KeyValue<TKey, TValue>

{

public TKey Key { get; set; }

public TValue Value { get; set; }

public KeyValue(TKey key, TValue value)

{

Key = key;

Value = value;

}

public static bool operator ==(KeyValue<TKey, TValue> item1, KeyValue<TKey, TValue> item2)

{

if(item1.Key.Equals(item2.Key) && item1.Value.Equals(item2.Value))

{

return true;

}

return false;

}

public static bool operator !=(KeyValue<TKey, TValue> item1, KeyValue<TKey, TValue> item2)

{

return !(item1 == item2);

}

public override int GetHashCode()

{

return Key.GetHashCode() + Value.GetHashCode();

}

public override bool Equals(object obj)

{

if (obj.GetType() != GetType()) return false;

KeyValue<TKey, TValue> item = (KeyValue<TKey, TValue>)obj;

return Key.Equals(item.Key) && Value.Equals(item.Value);

}

}

}

**ПРИЛОЖЕНИЕ Г**

Листинг класса *MainWindow.xaml.cs*.

using System;

using DictionaryLib;

using SubjectLib;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace DictionaryApplication

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

Dictionary<string, Subjects> timetable;

public MainWindow()

{

InitializeComponent();

DictionaryInitialization();

ListInitialization();

SomeValuesInitialization();

}

void DictionaryInitialization()

{

timetable = new Dictionary<string, Subjects>(new KeyValue<string, Subjects>[0]);

foreach(string weekDay in new string[] { "Понедельник", "Вторник", "Среда", "Четверг", "Пятница"})

{

timetable.Add(new KeyValue<string, Subjects>(weekDay, new Subjects()));

}

}

void ListInitialization()

{

timetableBox.Items.Clear();

foreach(KeyValue<string, Subjects> keyValue in timetable)

{

timetableBox.Items.Add(keyValue.Key);

for (int i = 0; i < keyValue.Value.Length; i++)

{

timetableBox.Items.Add(" " + keyValue.Value[i][0] + ": " + keyValue.Value[i][5]

+ " " + keyValue.Value[i][4] + " " + keyValue.Value[i][6]);

}

}

}

void SomeValuesInitialization()

{

timetable["Понедельник"].AddSubject("Математика", 120, 120, "Зачёт", "Александр",

"Бабич", "Антонович");

timetable["Среда"].AddSubject("ООП", 120, 120, "Зачёт", "Константин",

"Курочка", "Сергеевич");

timetable["Пятница"].AddSubject("Базы данных", 120, 120, "Зачёт", "Олег",

"Асенчик", "Даниилович");

ListInitialization();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

timetable[weekDayBox.Text].AddSubject(subjectNameBox.Text,

int.Parse(lecHoursBox.Text), int.Parse(labHoursBox.Text),

controlTypeBox.Text, eduFirstNameBox.Text, eduLastNameBox.Text,

eduMiddleNameBox.Text);

ListInitialization();

}

}

}