Текст программы в лабораторной работе №11

Класс Program

using System;

using System.Collections;

using System.Collections.Generic;

using LibraryLaba10;

namespace laba11

{

class Program

{

static void Main(string[] args)

{

int choiceMainMenu;

do

{

Menu.MainMenu();

choiceMainMenu = Check.EnterInt(1, 4, "Выберите пункт меню: ");

string keys = "";

switch (choiceMainMenu)

{

case 1:

{

Hashtable collHash = new Hashtable(); //создание хеш-таблицы

//добавление объектов созданной иерархии в хеш-таблицу

collHash.Add("t1", (Test)new Test().Init());

collHash.Add("t2", (Test)new Test().Init());

collHash.Add("e1", (Exam)new Exam().Init());

collHash.Add("e2", (Exam)new Exam().Init());

collHash.Add("fe1", (FinalExam)new FinalExam().Init());

collHash.Add("fe2", (FinalExam)new FinalExam().Init());

//переменная для проверки на сортировку

bool isSorted = false;

int choiceHashMenu;

do

{

Menu.HashMenu();

choiceHashMenu = Check.EnterInt(1, 10, "Выберите пункт меню: ");

switch (choiceHashMenu)

{

case 1: //добавление новых объектов в хеш-таблицу

{

int numberAdd = Check.EnterInt(1, 10, "Укажите количество добавляемых элементов: ");

for (int i = 0; i < numberAdd; i++)

{

string addObject = Check.EnterString("Укажите какой объект вы хотите добавить ? (Test / Exam / FinalExam) : ", "Такой функционал не осуществляется", "TestExamFinalExam");

switch (addObject)

{

case "Test":

string key = Check.EnterKey(collHash, "Укажите ключ объекта: ", "Введенный ключ уже используется");

collHash.Add(key, (Test)new Test().Init());

break;

case "Exam":

key = Check.EnterKey(collHash, "Укажите ключ объекта: ", "Введенный ключ уже используется");

collHash.Add(key, (Exam)new Exam().Init());

break;

case "FinalExam":

key = Check.EnterKey(collHash, "Укажите ключ объекта: ", "Введенный ключ уже используется");

collHash.Add(key, (FinalExam)new FinalExam().Init());

break;

}

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Объекты успешно добавлены");

Console.ResetColor();

break;

}

case 2: //удаление объектов из хеш-таблицы по ключам

{

if (!Check.IsEmpty(collHash))

{

int numberDel = Check.EnterInt(1, collHash.Count, "Укажите количество удаляемых элементов: ");

for (int i = 0; i < numberDel; i++)

{

Console.ResetColor();

string key = Check.EnterKey(1, collHash, "Укажите ключ объекта: ", "Объекта с таким ключом нет");

collHash.Remove(key);

if (keys != "")

{

int n = keys.IndexOf(key);

keys = keys.Remove(n, key.Length);

}

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Объекты успешно удалены");

Console.ResetColor();

}

break;

}

case 3: //запрос1: печать всех тестов

{

if (!Check.IsEmpty(collHash))

{

int count = 0;

foreach (var obj in collHash.Values)

{

if (obj is Test)

{

count++;

Console.WriteLine(((Test)obj).Print());

Console.WriteLine();

}

}

if (count == 0) Console.WriteLine("Экзамены отсутствуют");

}

break;

}

case 4: //запрос2: вывод среднего балла за все экзамены

{

if (!Check.IsEmpty(collHash))

{

int count = 0;

double sum = 0;

foreach (var obj in collHash.Values)

{

if (obj is Exam)

{

count++;

sum += ((Exam)obj).Grade;

}

}

if (count == 0) Console.WriteLine("Количество экзменов = 0, средний балл не посчитан");

else Console.WriteLine($"Средний балл за экзамены = {sum / count}");

}

break;

}

case 5: //запрос3: вывод кол-ва финальных экзаменов

{

if (!Check.IsEmpty(collHash))

{

int countFinalExams = 0;

foreach (var obj in collHash.Values)

{

if (obj is FinalExam)

{

countFinalExams++;

}

}

Console.WriteLine($"Количество финальных экзменов = {countFinalExams}");

}

break;

}

case 6: //вывод объектов хеш-таблицы на печать

{

if (!Check.IsEmpty(collHash))

{

Console.WriteLine("Получена коллекция: ");

Console.WriteLine();

foreach (var obj in collHash.Values)

{

Console.WriteLine(((Challenge)obj).Print());

Console.WriteLine();

}

}

break;

}

case 7: //клонирование хеш-таблицы

{

if (!Check.IsEmpty(collHash))

{

Hashtable collHashClone = (Hashtable)collHash.Clone();

collHashClone.Add("clone", (FinalExam)new FinalExam().Init());

Console.WriteLine("КЛОН");

foreach (var obj in collHashClone.Values)

{

Console.WriteLine(((Challenge)obj).Print());

Console.WriteLine();

}

Console.WriteLine($"Количество ссылок в клоне: {collHashClone.Count}");

Console.WriteLine("");

Console.WriteLine("ИЗНАЧАЛЬНАЯ КОЛЛЕКЦИЯ");

foreach (var obj in collHash.Values)

{

Console.WriteLine(((Challenge)obj).Print());

Console.WriteLine();

}

}

break;

}

case 8: //сортировка хеш-таблицы

{

if (!Check.IsEmpty(collHash))

{

if (isSorted)

{

Console.WriteLine("Коллекция уже отсортирована");

}

else

{

Challenge[] sortedArray = new Challenge[collHash.Count];

int index = 0;

foreach (var obj in collHash.Values)

{

sortedArray[index] = (Challenge)obj;

index++;

}

Array.Sort(sortedArray);

isSorted = true;

foreach (var obj in sortedArray)

{

Console.WriteLine(obj.Print());

Console.WriteLine();

}

}

}

break;

}

case 9: //поиск объекта в хеш-таблице по заданному ключу

{

if (!Check.IsEmpty(collHash))

{

string keyObj;

Console.WriteLine(keys);

keyObj = Check.EnterKey(1, collHash, "Укажите ключ объекта: ", "Объекта с таким ключом нет");

Console.WriteLine("Найденный объект: \n");

Console.WriteLine(((Challenge)collHash[keyObj]).Print());

Console.WriteLine("");

}

break;

}

case 10: Console.Clear(); break;

}

} while (choiceHashMenu < 10);

break;

}

case 2:

{

Queue<Challenge> collQueue = new Queue<Challenge>(); //создание очереди

//добавление объектов созданной иерархии в очередь

collQueue.Enqueue((Test)new Test().Init());

collQueue.Enqueue((Test)new Test().Init());

collQueue.Enqueue((Exam)new Exam().Init());

collQueue.Enqueue((Exam)new Exam().Init());

collQueue.Enqueue((FinalExam)new FinalExam().Init());

collQueue.Enqueue((FinalExam)new FinalExam().Init());

//переменная для проверки на сортировку

bool isSorted = false;

int choiceQueueMenu;

do

{

Menu.QueueMenu();

choiceQueueMenu = Check.EnterInt(1, 10, "Выберите пункт меню: ");

switch (choiceQueueMenu)

{

case 1: //добавление новых объектов в очередь

{

int numberAdd = Check.EnterInt(1, 10, "Укажите количество добавляемых элементов: ");

for (int i = 0; i < numberAdd; i++)

{

string addObject = Check.EnterString("Укажите какой объект вы хотите добавить ? (Test / Exam / FinalExam) : ", "Такой функционал не осуществляется", "TestExamFinalExam");

switch (addObject)

{

case "Test":

collQueue.Enqueue((Test)new Test().Init());

break;

case "Exam":

collQueue.Enqueue((Exam)new Exam().Init());

break;

case "FinalExam":

collQueue.Enqueue((FinalExam)new FinalExam().Init());

break;

}

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Объекты успешно добавлены");

Console.ResetColor();

break;

}

case 2: //удаление объектов из очереди

{

if (!Check.IsEmpty(collQueue))

{

int numberDel = Check.EnterInt(1, collQueue.Count, "Укажите количество удаляемых элементов: ");

for (int i = 0; i < numberDel; i++)

{

Console.ResetColor();

collQueue.Dequeue();

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Объекты успешно удалены");

Console.ResetColor();

}

break;

}

case 3: //запрос1: печать всех экзаменов

{

if (!Check.IsEmpty(collQueue))

{

int count = 0;

foreach (var obj in collQueue)

{

if (obj is Exam)

{

count++;

Console.WriteLine(((Exam)obj).Print());

Console.WriteLine();

}

}

if (count == 0) Console.WriteLine("Экзамены отсутствуют");

}

break;

}

case 4: //запрос2: вывод среднего балла за финальные экзамены

{

if (!Check.IsEmpty(collQueue))

{

int count = 0;

double sum = 0;

foreach (var obj in collQueue)

{

if (obj is FinalExam)

{

count++;

sum += ((FinalExam)obj).Grade;

}

}

if (count == 0) Console.WriteLine("Количество финальных экзменов = 0, средний балл не посчитан");

else Console.WriteLine($"Средний балл за экзамены = {sum / count}");

}

break;

}

case 5: //запрос3: вывод кол-ва тестов

{

if (!Check.IsEmpty(collQueue))

{

int countTests = 0;

foreach (var obj in collQueue)

{

if (obj is Test)

{

countTests++;

}

}

Console.WriteLine($"Количество тестов = {countTests}");

}

break;

}

case 6: //вывод очереди на печать

{

if (!Check.IsEmpty(collQueue))

{

Console.WriteLine("Получена коллекция:");

Console.WriteLine();

foreach (var obj in collQueue)

{

Console.WriteLine(((Challenge)obj).Print());

Console.WriteLine();

}

}

break;

}

case 7: //клонирование очереди

{

if (!Check.IsEmpty(collQueue))

{

Queue<Challenge> collQueueClone = new Queue<Challenge>(collQueue);

collQueueClone.Enqueue((FinalExam)new FinalExam().Init());

Console.WriteLine("КЛОН");

foreach (var obj in collQueueClone)

{

Console.WriteLine(((Challenge)obj).Print());

Console.WriteLine();

}

Console.WriteLine($"Количество ссылок в клоне: {collQueueClone.Count}");

Console.WriteLine("");

Console.WriteLine("ИЗНАЧАЛЬНАЯ КОЛЛЕКЦИЯ");

foreach (var obj in collQueue)

{

Console.WriteLine(((Challenge)obj).Print());

Console.WriteLine();

}

}

break;

}

case 8: //сортировка очереди

{

if (!Check.IsEmpty(collQueue))

{

if (isSorted)

{

Console.WriteLine("Коллекция уже отсортирована");

}

else

{

Challenge[] sortedArray = new Challenge[collQueue.Count];

int index = 0;

foreach (var obj in collQueue)

{

sortedArray[index] = (Challenge)obj;

index++;

}

Array.Sort(sortedArray);

isSorted = true;

foreach (var obj in sortedArray)

{

Console.WriteLine(obj.Print());

Console.WriteLine();

}

}

}

break;

}

case 9: //поиск объекта в очереди по заданному номеру

{

if (!Check.IsEmpty(collQueue))

{

int findObj = Check.EnterInt(1, collQueue.Count, "Введите номер объекта: ");

Console.WriteLine("");

Console.WriteLine((collQueue.ToArray())[findObj - 1].Print());

Console.WriteLine("");

}

break;

}

case 10: Console.Clear(); break;

}

} while (choiceQueueMenu < 10);

break;

}

case 3:

{

TestCollections tc = new TestCollections();

break;

}

}

} while (choiceMainMenu < 4);

}

}

}

Класс TestCollections

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LibraryLaba10;

using System.Diagnostics;

namespace laba11

{

class TestCollections

{

Stack<Challenge> collStack1;

Stack<string> collStack2;

Dictionary<Challenge, Test> collDict1;

Dictionary<string, Test> collDict2;

Stopwatch sw = new Stopwatch();

Challenge first;

Test firstValue;

Challenge middle;

Test middleValue;

Challenge last;

Test lastValue;

Challenge outof = new Challenge();

Test outofValue = new Test();

public TestCollections()

{

collStack1 = new Stack<Challenge>(1000);

collStack2 = new Stack<string>(1000);

collDict1 = new Dictionary<Challenge, Test>(1000);

collDict2 = new Dictionary<string, Test>(1000);

for (int i = 0; i < 1000; i++)

{

Test t = (Test)(new Test()).Init();

collStack1.Push(t.GetBase);

collStack2.Push(t.GetBase.ToString());

try

{

collDict1.Add(t.GetBase, t);

collDict2.Add(t.GetBase.ToString(), t);

}

catch

{

}

if (i == 0)

{

first = t.GetBase;

firstValue = t.GetTest;

}

if (i == 500)

{

middle = t.GetBase;

middleValue = t.GetTest;

}

if (i == 999)

{

last = t.GetBase;

lastValue = t.GetTest;

}

}

Menu();

}

private void Menu()

{

int choiceTestColl;

do

{

Console.WriteLine("\n1. Добавление объектов" +

"\n2. Удаление объектов" +

"\n3. Просмотр всех ключей" +

"\n4. Посмотреть затраченное время на поиск" +

"\n5. Назад" +

"\n");

choiceTestColl = Check.EnterInt(1, 5, "Выберите пункт меню: ");

switch (choiceTestColl)

{

case 1:

Console.WriteLine("\n1. Добавить объекты с помощью ДСЧ" +

"\n2. Добавить объекты с клавиатуры" +

"\n3. Назад");

int choiceAdd = Check.EnterInt(1, 3, "Выберите пункт меню: ");

int numberAdd = Check.EnterInt(1, 10, "Укажите количество добавляемых элементов: ");

switch (choiceAdd)

{

case 1:

for (int i = 0; i < numberAdd; i++)

{

Test t = (Test)new Test().Init();

Challenge c = t.GetBase;

collStack1.Push(c);

collStack2.Push(c.ToString());

try

{

collDict1.Add(c, t);

collDict2.Add(c.ToString(), t);

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[Ошибка!] Объект с таким ключом уже есть в коллекции");

Console.ResetColor();

}

}

break;

case 2:

for (int i = 0; i < numberAdd; i++)

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write("Введите тему теста: ");

Console.ResetColor();

string tp = Console.ReadLine();

int gr = Check.EnterInt(0, 10, "Введите оценку за тест: ");

Test t = new Test(gr, tp);

Challenge c = t.GetBase;

collStack1.Push(c);

collStack2.Push(c.ToString());

try

{

collDict1.Add(c, t);

collDict2.Add(c.ToString(), t);

}

catch

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[Ошибка!] Объект с таким ключом уже есть в коллекции");

Console.ResetColor();

}

}

break;

}

break;

case 2:

int numberDel = Check.EnterInt(1, collStack1.Count, "Укажите количество удаляемых элементов: ");

for (int i = 0; i < numberDel; i++)

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write("Введите тему теста: ");

Console.ResetColor();

string tp = Console.ReadLine();

int gr = Check.EnterInt(0, 10, "Введите оценку за тест: ");

int index = 0;

var b = (collDict1.Values).ToArray();

foreach (var obj in b)

{

if ((tp == b[index].Topic) && (gr == b[index].Grade))

{

b[index] = null;

}

index++;

}

index = 0;

collStack1.Clear();

collStack2.Clear();

collDict1.Clear();

collDict2.Clear();

foreach (var obj in b)

{

if (obj != null)

{

Challenge newB = obj.GetBase;

string newA = newB.ToString();

collStack1.Push(newB);

collStack2.Push(newA);

try

{

collDict1.Add(newB, obj);

collDict2.Add(newA, obj);

}

catch

{

}

}

}

}

break;

case 3:

foreach (var obj in collStack2)

Console.WriteLine(obj);

break;

case 4:

Console.Clear();

Console.WriteLine("Поиск первого элемента");

sw.Start();

if (collStack1.Contains(first))

{

sw.Stop();

Console.WriteLine($"Stack<Challenge> : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collStack2.Contains(first.ToString()))

{

sw.Stop();

Console.WriteLine($"Stack<string> : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collDict1.ContainsKey(first))

{

sw.Stop();

Console.WriteLine($"Dictionary<Challenge, Test> (ключ): {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collDict1.ContainsValue(firstValue))

{

sw.Stop();

Console.WriteLine($"Dictionary<Challenge, Test> (значение) : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collDict2.ContainsKey(first.ToString()))

{

sw.Stop();

Console.WriteLine($"Dictionary<string, Test> : {sw.Elapsed}");

}

sw.Reset();

Console.WriteLine();

Console.WriteLine("Поиск срединного элемента");

sw.Start();

if (collStack1.Contains(middle))

{

sw.Stop();

Console.WriteLine($"Stack<Challenge> : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collStack2.Contains(middle.ToString()))

{

sw.Stop();

Console.WriteLine($"Stack<string> : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collDict1.ContainsKey(middle))

{

sw.Stop();

Console.WriteLine($"Dictionary<Challenge, Test> (ключ) : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collDict1.ContainsValue(middleValue))

{

sw.Stop();

Console.WriteLine($"Dictionary<Challenge, Test> (значение) : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collDict2.ContainsKey(middle.ToString()))

{

sw.Stop();

Console.WriteLine($"Dictionary<string, Test> : {sw.Elapsed}");

}

sw.Reset();

Console.WriteLine();

Console.WriteLine("Поиск последнего элемента");

sw.Start();

if (collStack1.Contains(last))

{

sw.Stop();

Console.WriteLine($"Stack<Challenge> : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collStack2.Contains(last.ToString()))

{

sw.Stop();

Console.WriteLine($"Stack<string> : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collDict1.ContainsKey(last))

{

sw.Stop();

Console.WriteLine($"Dictionary<Challenge, Test> (ключ) : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collDict1.ContainsValue(lastValue))

{

sw.Stop();

Console.WriteLine($"Dictionary<Challenge, Test> (значение) : {sw.Elapsed}");

}

sw.Reset();

sw.Start();

if (collDict2.ContainsKey(last.ToString()))

{

sw.Stop();

Console.WriteLine($"Dictionary<string, Test> : {sw.Elapsed}");

}

sw.Reset();

Console.WriteLine();

Console.WriteLine("Поиск элемента не из коллекции");

sw.Start();

if (collStack1.Contains(outof))

{

sw.Stop();

Console.WriteLine($"Stack<Challenge> : {sw.Elapsed}");

}

else

Console.WriteLine("Не найден!");

sw.Reset();

sw.Start();

if (collStack2.Contains(outof.ToString()))

{

sw.Stop();

Console.WriteLine($"Stack<string> : {sw.Elapsed}");

}

else

Console.WriteLine("Не найден!");

sw.Reset();

sw.Start();

if (collDict1.ContainsKey(outof))

{

sw.Stop();

Console.WriteLine($"Dictionary<Challenge, Test> (ключ): {sw.Elapsed}");

}

else

Console.WriteLine("Не найден!");

sw.Reset();

sw.Start();

if (collDict1.ContainsValue(outofValue))

{

sw.Stop();

Console.WriteLine($"Dictionary<Challenge, Test> (значение) : {sw.Elapsed}");

}

else

Console.WriteLine("Не найден!");

sw.Reset();

sw.Start();

if (collDict2.ContainsKey(outof.ToString()))

{

sw.Stop();

Console.WriteLine($"Dictionary<string, Test> : {sw.Elapsed}");

}

else

Console.WriteLine("Не найден!");

sw.Reset();

Console.WriteLine();

break;

case 5: Console.Clear(); break;

}

} while (choiceTestColl < 5);

}

}

}

Класс Check

using LibraryLaba10;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace laba11

{

class Check

{

#region Ввод

public static int EnterInt(int minValue, int maxValue, string msg)

{

int intNum = 0;

bool isCorrect;

do

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write(msg);

Console.ResetColor();

try

{

intNum = Convert.ToInt32(Console.ReadLine());

isCorrect = true;

if ((intNum < minValue) || (intNum > maxValue))

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[Ошибка!] Выход за границы допустимых значений");

Console.ResetColor();

isCorrect = false;

}

}

catch (FormatException)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[Ошибка!] Нужно вводить только целые числа");

Console.ResetColor();

isCorrect = false;

}

catch (OverflowException)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[Ошибка!] Введено слишком большое число");

Console.ResetColor();

isCorrect = false;

}

} while (!isCorrect);

return intNum;

}

public static string EnterString(string msg, string msgError, string checkString)

{

string str;

bool isCorrect;

do

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write(msg);

Console.ResetColor();

str = Console.ReadLine();

if (checkString.Contains(str))

{

isCorrect = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[Ошибка!] " + msgError);

Console.ResetColor();

isCorrect = false;

}

} while (!isCorrect);

return str;

}

public static string EnterKey(Hashtable hsht, string msg, string msgError)

{

string key;

bool isCorrect;

do

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write(msg);

Console.ResetColor();

key = Console.ReadLine();

if (hsht.ContainsKey(key))

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[Ошибка!] " + msgError);

Console.ResetColor();

isCorrect = false;

}

else

{

isCorrect = true;

}

} while (!isCorrect);

return key;

}

public static string EnterKey(int operation, Hashtable hsht, string msg, string msgError)

{

string key;

bool isCorrect;

do

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write(msg);

Console.ResetColor();

key = Console.ReadLine();

if (hsht.ContainsKey(key))

{

isCorrect = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[Ошибка!] " + msgError);

Console.ResetColor();

isCorrect = false;

}

} while (!isCorrect);

return key;

}

public static bool IsEmpty(Hashtable hsht)

{

bool isEmpty;

if (hsht == null || hsht.Count == 0)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[Ошибка!] Коллекция пуста");

Console.ResetColor();

isEmpty = true;

}

else

{

isEmpty = false;

}

return isEmpty;

}

public static bool IsEmpty(Queue<Challenge> que)

{

bool isEmpty;

if (que == null || que.Count == 0)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[Ошибка!] Коллекция пуста");

Console.ResetColor();

isEmpty = true;

}

else

{

isEmpty = false;

}

return isEmpty;

}

#endregion

}

}

Класс Menu

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace laba11

{

class Menu

{

#region Главное меню ЛР11

public static void MainMenu()

{

Console.WriteLine("1. Задание №1" +

"\n2. Задание №2" +

"\n3. Задание №3" +

"\n4. Завершение работы" +

"\n");

}

#endregion

#region Меню для хеш-таблиц (задание №1)

public static void HashMenu()

{

Console.WriteLine("\n1. Добавление объектов" +

"\n2. Удаление объектов" +

"\n3. Запросить печать всех тестов" +

"\n4. Запросить средний балл за экзамены" +

"\n5. Запросить количество финальных экзаменов" +

"\n6. Вывести все объекты коллекции на печать " +

"\n7. Выполнить клонирование коллекции" +

"\n8. Сортировка по имени испытания" +

"\n9. Поиск объекта в коллекции" +

"\n10. Назад" +

"\n");

}

#endregion

#region Меню для очереди (задание №1)

public static void QueueMenu()

{

Console.WriteLine("\n1. Добавление объектов" +

"\n2. Удаление объектов" +

"\n3. Запросить печать всех экзаменов" +

"\n4. Запросить средний балл за финальные экзамены" +

"\n5. Запросить количество тестов" +

"\n6. Вывести все объекты коллекции на печать " +

"\n7. Выполнить клонирование коллекции" +

"\n8. Сортировка по имени испытания" +

"\n9. Поиск объекта в коллекции" +

"\n10. Назад" +

"\n");

}

#endregion

}

}