Лабораторная работа 9 по С#

Обобщения и коллекции.

студент, заочное, 40331, Василевский Евгений Валерьевич

**Задание 1**

Задание 1. Непараметризованные коллекции Создайте непараметризованную коллекцию указанного типа. Заполните ее 10 значениями различных типов (int, double, float, enum, string, DateTime, object, объекты ваших классов, кортежи, массивы и т.д.). Выведите коллекцию на консоль. Выберите элементы коллекции, соответствующие условию, и переместите их в другую коллекцию. Выведите исходную и полученную коллекции на консоль.

internal class Program

{

enum Example { one, two, three };

private static void Main(string[] args)

{

Console.WriteLine("Лабораторная работа 9. Задание 1");

Console.WriteLine();

//Вариант 1. Исходная коллекция ArrayList. Выберите числовые значения от 10

//до 20. Поместите в Hashtable.

ArrayList list = new ArrayList();

list.Add(12); //int

list.Add(3.14); //double

list.Add("Hello, World!"); //string

list.Add(DateTime.Now); //DateTime

list.Add(Example.one); //enum

list.Add(5.5f); //float

list.Add(DayOfWeek.Monday); //enum

list.Add(new object()); //object

list.Add(new Person { Name = "Alice", Age = 30 }); //объект класса

list.Add(new int[] { 1, 2, 3 }); //массив

list.Add("Another string"); //string

list.Add(42); //int

list.Add(2.718); //double

list.Add(true); //bool

list.Add(new float[] { 1.1f, 2.2f }); //массив

Hashtable ht = new Hashtable();

foreach(var item in list)

{

if(item is int)

{

if (Convert.ToInt32(item) >= 10 && Convert.ToInt32(item) <= 20)

ht.Add(item.GetHashCode(), item);

}

}

foreach(var item in ht)

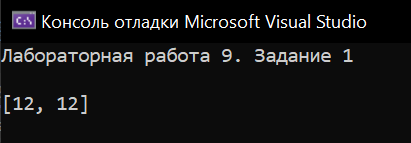
{

Console.WriteLine($"{item}");

}

}

}



//Вариант 3. Исходная коллекция Queue. Выберите только классы. Поместите в

//ArrayList.

ArrayList list = new ArrayList();

Queue queue = new Queue();

queue.Enqueue(12); //int

queue.Enqueue(3.14); //double

queue.Enqueue("Hello, World!"); //string

queue.Enqueue(DateTime.Now); //DateTime

queue.Enqueue(Example.one); //enum

queue.Enqueue(5.5f); //float

queue.Enqueue(DayOfWeek.Monday); //enum

queue.Enqueue(new object()); //object

queue.Enqueue(new Person { Name = "Alice", Age = 30 }); //объект класса

queue.Enqueue(new int[] { 1, 2, 3 }); //массив

queue.Enqueue("Another string"); //string

queue.Enqueue(42); //int

queue.Enqueue(2.718); //double

queue.Enqueue(true); //bool

queue.Enqueue(new float[] { 1.1f, 2.2f }); //массив

foreach (var item in queue)

{

if(item is Person)

{

list.Add(item);

}

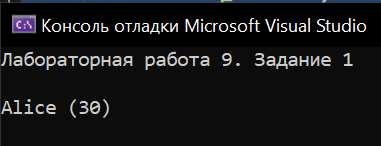
}

foreach (var item in list)

{

Console.WriteLine(item);

}



//Вариант 4.Исходная коллекция Hashtable. Выберите только перечисления.

//Поместите в Stack.

Hashtable hashtable = new Hashtable();

Stack stack = new Stack();

ArrayList list = new ArrayList();

list.Add(12); //int

list.Add(3.14); //double

list.Add("Hello, World!"); //string

list.Add(DateTime.Now); //DateTime

list.Add(Example.one); //enum

list.Add(5.5f); //float

list.Add(DayOfWeek.Monday); //enum

list.Add(new object()); //object

list.Add(new Person { Name = "Alice", Age = 30 }); //объект класса

list.Add(new int[] { 1, 2, 3 }); //массив

list.Add("Another string"); //string

list.Add(42); //int

list.Add(2.718); //double

list.Add(true); //bool

list.Add(new float[] { 1.1f, 2.2f }); //массив

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

{

hashtable.Add(i, list[i]);

}

foreach(DictionaryEntry i in hashtable)

{

if(i.Value is Enum)

{

stack.Push(i.Value);

}

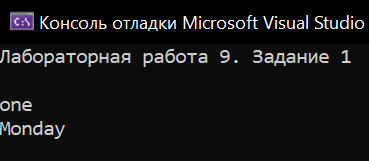
}

foreach (var i in stack)

{

Console.WriteLine(i);

}



//Вариант 5. Исходная коллекция Sortedlist. Выберите только отрицательные

//вещественные числа. Поместите в Queue.

ArrayList list = new ArrayList();

list.Add(12); //int

list.Add(-3.14); //double

list.Add("Hello, World!"); //string

list.Add(DateTime.Now); //DateTime

list.Add(Example.one); //enum

list.Add(5.5f); //float

list.Add(DayOfWeek.Monday); //enum

list.Add(new object()); //object

list.Add(new Person { Name = "Alice", Age = 30 }); //объект класса

list.Add(new int[] { 1, 2, 3 }); //массив

list.Add("Another string"); //string

list.Add(42); //int

list.Add(2.718); //double

list.Add(true); //bool

list.Add(new float[] { 1.1f, 2.2f }); //массив

SortedList sortedList = new SortedList();

Queue queue = new Queue();

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

{

sortedList.Add(i, list[i]);

}

foreach(DictionaryEntry entry in sortedList)

{

if(entry.Value is Double || entry.Value is Single)

{

if (Convert.ToDouble(entry.Value) < 0)

{

queue.Enqueue(entry.Value);

}

}

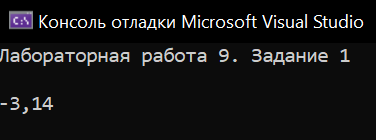
}

foreach(var item in queue)

{

Console.WriteLine(item);

}



//Вариант 6.Исходная коллекция ArrayList. Выберите только массивы длиной

//свыше 3 элементов.Поместите в Hashtable.

ArrayList list = new ArrayList();

list.Add(12); //int

list.Add(3.14); //double

list.Add("Hello, World!"); //string

list.Add(DateTime.Now); //DateTime

list.Add(Example.one); //enum

list.Add(5.5f); //float

list.Add(DayOfWeek.Monday); //enum

list.Add(new object()); //object

list.Add(new Person { Name = "Alice", Age = 30 }); //объект класса

list.Add(new int[] { 1, 2, 3 }); //массив

list.Add("Another string"); //string

list.Add(42); //int

list.Add(2.718); //double

list.Add(true); //bool

list.Add(new float[] { 1.1f, 2.2f }); //массив

Hashtable ht = new Hashtable();

int i = 0;

foreach (var item in list)

{

if (item is Array array && array.Length>=3)

{

ht.Add(i, item);

i++;

}

}

foreach (DictionaryEntry item in ht)

{

Array array1=item.Value as Array;

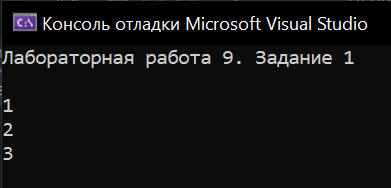
foreach(var item2 in array1)

{

Console.WriteLine(item2);

}

}



//Вариант 7. Исходная коллекция Stack. Выберите только строки длиной более 3.

//Поместите в Sortedlist.

Stack stack = new Stack();

stack.Push(10); //int

stack.Push(3.14); //double

stack.Push("Hello, World!"); //string

stack.Push(DateTime.Now); //DateTime

stack.Push(Example.one); //enum

stack.Push(5.5f); //float

stack.Push(DayOfWeek.Monday); //enum

stack.Push(new object()); //object

stack.Push(new Person { Name = "Alice", Age = 30 }); //объект класса

stack.Push(new int[] { 1, 2, 3 }); //массив

stack.Push("Another string"); //string

stack.Push(42); //int

stack.Push(2.718); //double

stack.Push(true); //bool

stack.Push(new float[] { 1.1f, 2.2f }); //массив

SortedList sortedList = new SortedList();

foreach (var item in stack)

{

Console.WriteLine(item);

}

Console.WriteLine();

Console.WriteLine("------------- SortedList---------------");

Console.WriteLine();

foreach (var item in stack)

{

if (item is String && Convert.ToString(item).Length>3)

{

sortedList.Add(item.GetHashCode(), item);

}

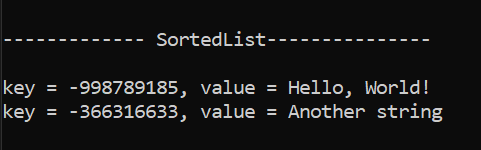
}

foreach (DictionaryEntry entry in sortedList)

{

Console.WriteLine($"key = {entry.Key}, value = {entry.Value}");

}



//Вариант 8. Исходная коллекция Queue. Выберите только положительные числа.

//Поместите в ArrayList.

ArrayList list = new ArrayList();

Queue queue = new Queue();

queue.Enqueue(-12); //int

queue.Enqueue(3.14); //double

queue.Enqueue("Hello, World!"); //string

queue.Enqueue(DateTime.Now); //DateTime

queue.Enqueue(Example.one); //enum

queue.Enqueue(5.5f); //float

queue.Enqueue(DayOfWeek.Monday); //enum

queue.Enqueue(new object()); //object

queue.Enqueue(new Person { Name = "Alice", Age = 30 }); //объект класса

queue.Enqueue(new int[] { 1, 2, 3 }); //массив

queue.Enqueue("Another string"); //string

queue.Enqueue(42); //int

queue.Enqueue(2.718); //double

queue.Enqueue(true); //bool

queue.Enqueue(new float[] { 1.1f, 2.2f }); //массив

foreach (var item in queue)

{

if (item is double || item is float || item is int && Convert.ToDouble(item)>0)

{

list.Add(item);

}

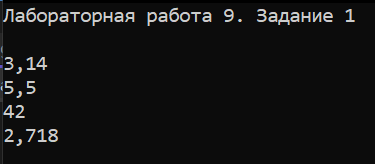
}

foreach (var item in list)

{

Console.WriteLine(item);

}



//Вариант 9. Исходная коллекция Hashtable. Выберите только даты. Поместите в

//Stack.

Hashtable hashtable = new Hashtable();

Stack stack = new Stack();

ArrayList list = new ArrayList();

list.Add(12); //int

list.Add(3.14); //double

list.Add("Hello, World!"); //string

list.Add(DateTime.Now); //DateTime

list.Add(Example.one); //enum

list.Add(5.5f); //float

list.Add(DayOfWeek.Monday); //enum

list.Add(new object()); //object

list.Add(new Person { Name = "Alice", Age = 30 }); //объект класса

list.Add(new int[] { 1, 2, 3 }); //массив

list.Add("Another string"); //string

list.Add(42); //int

list.Add(2.718); //double

list.Add(true); //bool

list.Add(new float[] { 1.1f, 2.2f }); //массив

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

{

hashtable.Add(i, list[i]);

}

foreach (DictionaryEntry i in hashtable)

{

if (i.Value is DateTime)

{

stack.Push(i.Value);

}

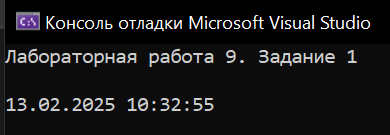
}

foreach (var i in stack)

{

Console.WriteLine(i);

}



//Вариант 10. Исходная коллекция Sortedlist. Выберите только строки, которые

//конвертируются в целые числа. Поместите в Queue.

ArrayList list = new ArrayList();

list.Add(12); //int

list.Add(-3.14); //double

list.Add("Hello, World!"); //string

list.Add(DateTime.Now); //DateTime

list.Add(Example.one); //enum

list.Add(5.5f); //float

list.Add(DayOfWeek.Monday); //enum

list.Add(new object()); //object

list.Add(new Person { Name = "Alice", Age = 30 }); //объект класса

list.Add(new int[] { 1, 2, 3 }); //массив

list.Add("Another string"); //string

list.Add(42); //int

list.Add(2.718); //double

list.Add(true); //bool

list.Add(new float[] { 1.1f, 2.2f }); //массив

SortedList sortedList = new SortedList();

Queue queue = new Queue();

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

{

sortedList.Add(i, list[i]);

}

foreach (DictionaryEntry entry in sortedList)

{

if (entry.Value is int)

{

queue.Enqueue(entry.Value);

}

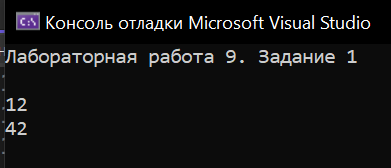
}

foreach (var item in queue)

{

Console.WriteLine(item);

}



Задание 2

Задание 2. ArrayList ArrayList содержит объекты различных типов. Сгруппировать объекты по различным коллекциям в зависимости от типа. Типы могут быть сложными (например, объекты ваших классов).

Console.WriteLine("Лабораторная работа 9. Задание 2");

ArrayList arrayList = new ArrayList();

arrayList.Add(1);

arrayList.Add(new Car { Name = "Audi"});

arrayList.Add(new Plane { Name = "Boing"});

arrayList.Add(DayOfWeek.Monday);

arrayList.Add(true);

arrayList.Add(6.7);

arrayList.Add(7.6f);

arrayList.Add("Hello, World!");

arrayList.Add(9);

arrayList.Add(new Car { Name = "BMW"});

arrayList.Add(11f);

arrayList.Add(new Plane { Name = "MIG-27"});

arrayList.Add('N');

arrayList.Add("Another string");

arrayList.Add(new Car { Name = "Mersedes"});

arrayList.Add(16.99);

arrayList.Add(false);

arrayList.Add(18);

arrayList.Add(new float[] { 1.1f, 2.2f });

arrayList.Add(-20);

foreach (var item in arrayList)

{

Console.WriteLine(item);

}

Console.WriteLine("----------------------------");

var dictionary = new Dictionary<string, ArrayList>();

foreach (var item in arrayList)

{

string typeName = item.GetType().Name;

//Console.WriteLine(typeName);

if (!dictionary.ContainsKey(typeName))

{

dictionary[typeName] = new ArrayList();

}

dictionary[typeName].Add(item);

}

foreach (var kv in dictionary)

{

Console.WriteLine($"Key:{kv.Key}");

foreach (var item in kv.Value)

{

Console.WriteLine(item.ToString());

}

Console.WriteLine();

}

public class Car

{

public Car()

{

}

public Car(string name)

{

Name = name;

}

public string Name { get; set; }

public override string? ToString()

{

return $"Машина марки {Name}";

}

public class Plane

{

public Plane()

{

}

public Plane(string name)

{

Name = name;

}

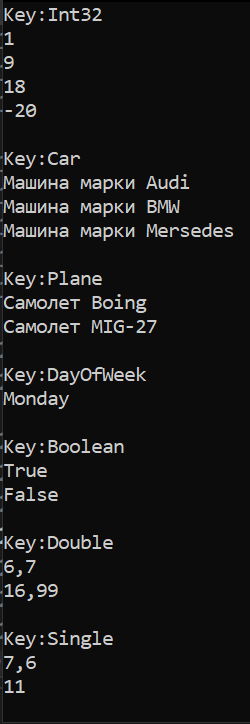
public string Name { get; set; }

public override string? ToString()

{

return $"Самолет {Name}";

}



Задание 3

Задание 3. Использование коллекции List. Используя List, создать коллекцию объектов любого базового класса из предыдущих работ. Сразу проинициализировать ее значениями различных классов.

Console.WriteLine("Лабораторная работа 9. Задание 3");

List<Auto> list = new List<Auto>();

list.Add(new Auto("BMW"));

list.Add(new Auto("Audi"));

list.Add(new Batmobile("Dodge","Batman"));

list.Add(new Auto("Mersedes"));

list.Add(new Auto("Renault"));

list.Add(new Auto("Maserati"));

list.Add(new Batmobile("Porshe","Robin"));

while (true)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* Главное меню \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("1 - просмотр коллекции");

Console.WriteLine("2 - добавление элемента");

Console.WriteLine("3 - добавление элемента по указанному индексу");

Console.WriteLine("4 - нахождение элемента с начала коллекции");

Console.WriteLine("5 - нахождение элемента с конца коллекции");

Console.WriteLine("6 - удаление элемента по индексу");

Console.WriteLine("7 - удаление элемента по значению");

Console.WriteLine("8 - реверс коллекции");

Console.WriteLine("9 - сортировка");

Console.WriteLine("10 - выполнение методов всех объектов, поддерживающих IFlying");

Console.WriteLine("0 - выход");

int otvet = Convert.ToInt32(Console.ReadLine());

switch (otvet)

{

case 1: method1(list); break;

case 2: method2(list); break;

case 3: method3(list); break;

case 4: method4(list); break;

case 5: method5(list); break;

case 6: method6(list); break;

case 7: method7(list); break;

case 8: method8(list); break;

case 9: method9(list); break;

case 10: method10(list); break;

default: return;

}

}

void method10(List<Auto> list)

{

//10 - выполнение методов всех объектов, поддерживающих IFlying

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

{

if (list[i] is IFlying)

{

((IFlying)list[i]).Fly();

}

}

Console.ReadKey();

}

void method9(List<Auto> list)

{

//9 - сортировка

list.Sort();

Console.WriteLine("Сортировка завершена");

Console.ReadKey();

}

void method8(List<Auto> list)

{

//8 - реверс коллекции

list.Reverse();

Console.WriteLine("Реверс завершен");

Console.ReadKey();

}

void method7(List<Auto> list)

{

//7 - удаление элемента по значению

Console.Write("name: ");

string name = Console.ReadLine();

Auto auto = new Auto(name);

if(list.Remove(auto))

Console.WriteLine("Элемент успешно удален");

else

{

Console.WriteLine("Элемент не найден");

}

Console.ReadKey();

}

void method6(List<Auto> list)

{

//6 - удаление элемента по индексу

Console.Write("индекс: ");

string otvet = Console.ReadLine();

int index;

if(!Int32.TryParse(otvet, out index) || index<0 || index >= list.Count)

{

Console.WriteLine("неверный индекс");

Console.ReadKey();

return;

}

list.RemoveAt(index);

Console.WriteLine("Элемент успешно удален");

Console.ReadKey();

}

void method5(List<Auto> list)

{

//5 - нахождение элемента с конца коллекции

Console.Write("name: ");

string name = Console.ReadLine();

Auto auto = new Auto(name);

int index = list.LastIndexOf(auto);

Console.WriteLine(index > -1 ? $"\nиндекс ={index}; {list[index]}" : "элемент не найден");

Console.ReadKey();

}

void method4(List<Auto> list)

{

//4-нахождение элемента с начала коллекции

Console.Write("name: ");

string name=Console.ReadLine();

Auto auto = new Auto(name);

int index=list.IndexOf(auto);

Console.WriteLine(index>-1 ? $"\nиндекс ={index}; {list[index]}":"элемент не найден");

Console.ReadKey();

}

void method3(List<Auto> list)

{

//3 - добавление элемента по указанному индексу

Console.Write("индекс: ");

string otvet = Console.ReadLine();

int index;

if (!Int32.TryParse(otvet, out index) || index < 0 || index >= list.Count)

{

Console.WriteLine("неверный индекс");

Console.ReadKey();

return;

}

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name = Console.ReadLine();

string driver = Console.ReadLine();

switch (n)

{

case 1: list.Insert(index,new Auto(name)); break;

case 2: list.Insert(index, new Batmobile(name,driver)); break;

}

Console.WriteLine("Элемент добавлен");

Console.ReadKey();

}

void method2(List<Auto> list)

{

//2 - добавление элемента

Console.WriteLine("Добавляем Auto или Batmobile (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("Если 1, то name:, если 2, то name: driver: ");

string name = Console.ReadLine();

string driver = Console.ReadLine();

switch (n)

{

case 1:list.Add(new Auto(name)); break;

case 2: list.Add(new Batmobile(name, driver)); break;

}

Console.WriteLine("Элемент добавлен");

Console.ReadKey();

}

void method1(List<Auto> list)

{

//1-просмотр коллекции

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

{

Console.WriteLine($"{i}) {list[i]}");

}

Console.ReadKey();

}

public class Auto:IComparable

{

private string name;

public Auto() : this("Noname")

{

}

public Auto(string name)

{

Name = name;

}

public string Name { get => name; set => name = value; }

public int CompareTo(object other)

{

return this.name.CompareTo((other as Auto)?.Name ?? "");

}

public override bool Equals(object? obj)

{

return obj is Auto auto && Name == auto.Name;

}

public override string? ToString()

{

return $"Я машина марки {name}";

}

}

internal class Batmobile:Auto,IFlying

{

private string driver;

public Batmobile() : this("Noname", "Бэтмен")

{

}

public Batmobile(string name,string driver):base(name)

{

Driver = driver;

}

public string Driver { get => driver; set => driver = value; }

public void Fly()

{

Console.WriteLine("Я бетмобиль - так что могу превращаться в самолет и летать");

}

public override string? ToString()

{

return $"Я бетмобиль, за рулем сидит {driver}";

}

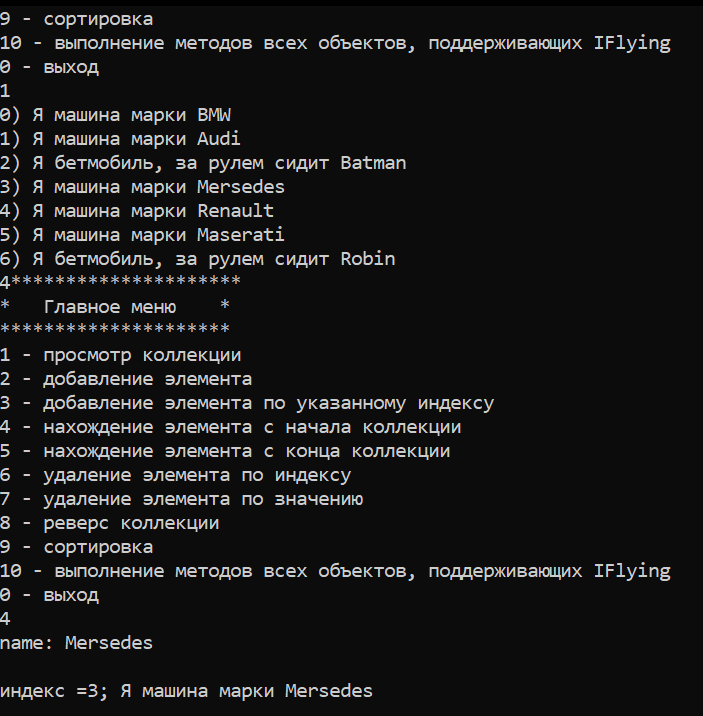
}

internal interface IFlying

{

void Fly();

}



Задание 4

Задание 4. Использование параметризованной коллекции Используя параметризованную коллекцию, продемонстрировать все ее возможности по аналогии с предыдущим заданием.

Dictionary (добавить работу с ключами)

Console.WriteLine("Лабораторная работа 9. Задание 4");

List<Pet> list = new List<Pet>();

list.Add(new Pet("хорек",2));

list.Add(new Pet("крокодил",1));

list.Add(new Bird("воробей", 2));

list.Add(new Pet("хорек",3));

list.Add(new Pet("волк",7));

list.Add(new Pet("лев",6));

list.Add(new Bird("синица", 5));

Dictionary<int,Pet> dict = new Dictionary<int,Pet>();

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

{

dict.Add(i, list[i]);

}

while (true)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* Главное меню \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("1 - просмотр коллекции");

Console.WriteLine("2 - добавление элемента");

Console.WriteLine("3 - очищение словаря");

Console.WriteLine("4 - пров. нал-е эл-та с опр. кл. и возвр. true при его нал-и в сл-ре");

Console.WriteLine("5 - пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в сл-ре");

Console.WriteLine("6 - удаление элемента по ключу");

Console.WriteLine("7 - удаление элемента по ключу и возвращает знач-е");

Console.WriteLine("8 - получает из словаря элемент по ключу key");

Console.WriteLine("9 - выполнение методов всех объектов, поддерживающих IFlying");

Console.WriteLine("0 - выход");

int otvet = Convert.ToInt32(Console.ReadLine());

switch (otvet)

{

case 1: method1(dict); break;

case 2: method2(dict); break;

case 3: method3(dict); break;

case 4: method4(dict); break;

case 5: method5(dict); break;

case 6: method6(dict); break;

case 7: method7(dict); break;

case 8: method8(dict); break;

case 9: method9(dict); break;

default: return;

}

}

void method9(Dictionary<int, Pet> dict)

{

//9 - выполнение методов всех объектов, поддерживающих IFlying

foreach(KeyValuePair<int, Pet> pair in dict)

{

if (pair.Value is IFlying)

{

((IFlying)pair.Value).Fly();

}

}

Console.ReadKey();

}

void method8(Dictionary<int, Pet> dict)

{

//8 - получает из словаря элемент по ключу key

Console.Write("ключ: ");

string otvet = Console.ReadLine();

int index;

if (!Int32.TryParse(otvet, out index) || index < 0 || index >= dict.Count)

{

Console.WriteLine("неверный индекс");

Console.ReadKey();

return;

}

Pet pet = new Pet();

Console.WriteLine(dict.TryGetValue(index, out pet)?"Элемент успешно найден":"элемент не найден");

Console.WriteLine($"найденный элемент: {pet}");

Console.ReadKey();

}

void method7(Dictionary<int, Pet> dict)

{

//7 - удаление элемента по ключу и возвращает знач-е

Console.Write("ключ: ");

string otvet = Console.ReadLine();

int index;

if (!Int32.TryParse(otvet, out index) || index < 0 || index >= list.Count)

{

Console.WriteLine("неверный индекс");

Console.ReadKey();

return;

}

Pet pet = new Pet();

dict.Remove(index,out pet);

Console.WriteLine("Элемент успешно удален");

Console.WriteLine(pet);

Console.ReadKey();

}

void method6(Dictionary<int, Pet> dict)

{

//6 - удаление элемента по ключу

Console.Write("ключ: ");

string otvet = Console.ReadLine();

int index;

if (!Int32.TryParse(otvet, out index) || index < 0 || index >= list.Count)

{

Console.WriteLine("неверный индекс");

Console.ReadKey();

return;

}

dict.Remove(index);

Console.WriteLine("Элемент успешно удален");

Console.ReadKey();

}

void method5(Dictionary<int, Pet> dict)

{

//5 - пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в сл-ре

Console.Write("Value: ");

string Value = Console.ReadLine();

Pet pet=new Pet(Value);

Console.WriteLine(dict.ContainsValue(pet) ? true : false);

foreach (KeyValuePair<int, Pet> pair in dict)

{

if (pair.Value.Name == pet.Name)

{

Console.WriteLine($"Key: {pair.Key}; Value: {pair.Value}");

}

}

Console.ReadKey();

}

void method4(Dictionary<int, Pet> dict)

{

//4 - пров. нал-е эл-та с опр. кл. и возвр. true при его нал-и в сл-ре

Console.Write("Key: ");

string Key = Console.ReadLine();

int index;

if(!Int32.TryParse(Key, out index)|| index<0 || index>=dict.Count)return;

Console.WriteLine(dict.ContainsKey(index)? true : false);

foreach(KeyValuePair<int, Pet> pair in dict)

{

if(pair.Key == index)

{

Console.WriteLine($"Key: {pair.Key}; Value: {pair.Value}");

}

}

Console.ReadKey();

}

void method3(Dictionary<int, Pet> dict)

{

//3 - очищение словаря

dict.Clear();

Console.WriteLine("Словарь очищен");

Console.ReadKey();

}

void method2(Dictionary<int,Pet> dict)

{

//2 - добавление элемента

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name = Console.ReadLine();

Console.Write("age: ");

otvet = Console.ReadLine();

if (!Int32.TryParse(otvet, out age) || age < 0 || age > 30) return;

switch (n)

{

case 1: dict.Add(dict.Count,new Pet(name,age)); break;

case 2: dict.Add(dict.Count, new Bird(name, age)); break;

}

Console.WriteLine("Элемент добавлен");

Console.ReadKey();

}

void method1(Dictionary<int,Pet> dict)

{

//1-просмотр коллекции

foreach(KeyValuePair<int,Pet> pair in dict)

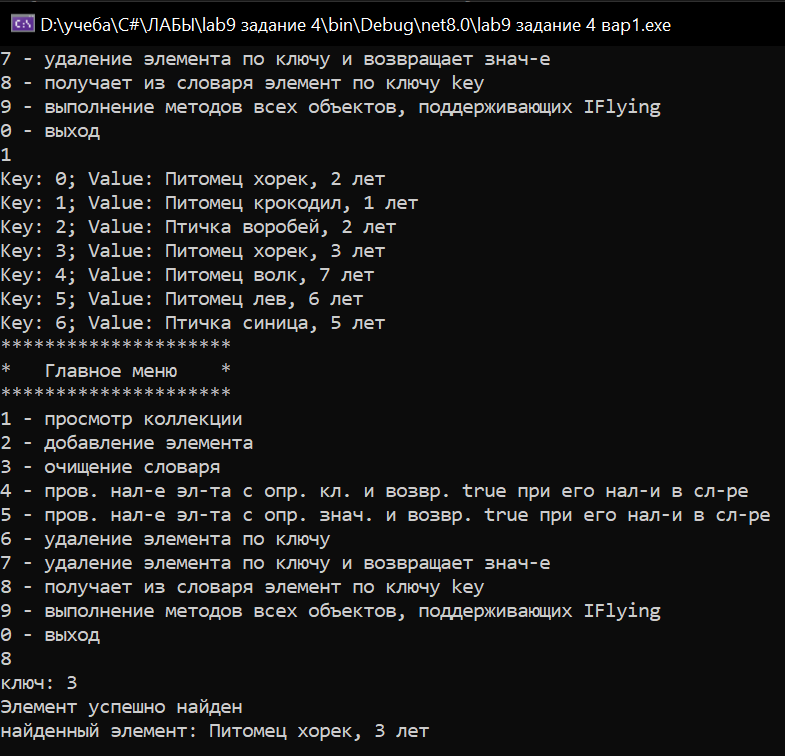
{

Console.WriteLine($"Key: {pair.Key}; Value: {pair.Value}");

}

Console.ReadKey();

}



Console.WriteLine("Лабораторная работа 9. Задание 4");

//LinkedList (добавить просмотр/добавление/удаление предыдущего/след.

//элемента)

LinkedList<Pet> list = new LinkedList<Pet>();

list.AddLast(new Pet("хорек", 2));

list.AddLast(new Pet("крокодил", 1));

list.AddLast(new Bird("воробей", 2));

list.AddLast(new Pet("хорек", 3));

list.AddLast(new Pet("волк", 7));

list.AddLast(new Pet("лев", 6));

list.AddLast(new Bird("синица", 5));

while (true)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* Главное меню \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("1 - просмотр коллекции");

Console.WriteLine("2 - доб-е эл-та в сп-к в конец списка");

Console.WriteLine("3 - добавление элемента в начало списка");

Console.WriteLine("4 - нахождение элемента с начала коллекции");

Console.WriteLine("5 - нахождение элемента с конца коллекции");

Console.WriteLine("6 - удаление элемента в начале списка");

Console.WriteLine("7 - удаление элемента в конце списка");

Console.WriteLine("8 - реверс коллекции");

Console.WriteLine("9 - выполнение методов всех объектов, поддерживающих IFlying");

Console.WriteLine("10 - просмотр след. элемента");

Console.WriteLine("11 - просмотр пред. элемента");

Console.WriteLine("12 - доб. след. элемента");

Console.WriteLine("13 - доб. пред. элемента");

Console.WriteLine("0 - выход");

int otvet = Convert.ToInt32(Console.ReadLine());

switch (otvet)

{

case 1: method1(list); break;

case 2: method2(list); break;

case 3: method3(list); break;

case 4: method4(list); break;

case 5: method5(list); break;

case 6: method6(list); break;

case 7: method7(list); break;

case 8: method8(list); break;

case 9: method9(list); break;

case 10: method10(list); break;

case 11: method11(list); break;

case 12: method12(list); break;

case 13: method13(list); break;

default: return;

}

}

void method13(LinkedList<Pet> list)

{

//10-доб. пред. элемента

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

var current = list.Last;

while (current != null)

{

if (current.Value.Equals(pet))

{

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name1 = Console.ReadLine();

Console.Write("age: ");

otvet = Console.ReadLine();

if (!Int32.TryParse(otvet, out age) || age < 0 || age > 30) return;

switch (n)

{

case 1: list.AddBefore(current, new Pet(name1, age)); break;

case 2: list.AddBefore(current, new Bird(name1, age)); break;

}

Console.WriteLine("Элемент добавлен");

Console.ReadKey();

return;

}

current = current.Previous;

}

Console.WriteLine("Элемент не найден");

Console.ReadKey();

}

void method12(LinkedList<Pet> list)

{

//10-доб. след. элемента

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

var current = list.First;

while (current != null)

{

if (current.Value.Equals(pet))

{

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name1 = Console.ReadLine();

Console.Write("age: ");

otvet = Console.ReadLine();

if (!Int32.TryParse(otvet, out age) || age < 0 || age > 30) return;

switch (n)

{

case 1: list.AddAfter(current,new Pet(name1, age)); break;

case 2: list.AddAfter(current,new Bird(name1, age)); break;

}

Console.WriteLine("Элемент добавлен");

Console.ReadKey();

return;

}

current = current.Next;

}

Console.WriteLine("Элемент не найден");

Console.ReadKey();

}

void method11(LinkedList<Pet> list)

{

//11-просмотр пред. элемента

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

var current = list.Last;

while (current != null)

{

if (current.Value.Equals(pet))

{

Console.WriteLine($"Предыдущий элемент: {current.Previous.Value}");

Console.ReadKey();

return;

}

current = current.Previous;

}

Console.WriteLine("Элемент не найден");

Console.ReadKey();

}

void method10(LinkedList<Pet> list)

{

//10-просмотр след. элемента

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

var current = list.First;

while(current != null)

{

if (current.Value.Equals(pet))

{

Console.WriteLine($"Следующий элемент: {current.Next.Value}");

Console.ReadKey();

return;

}

current = current.Next;

}

Console.WriteLine("Элемент не найден");

Console.ReadKey();

}

void method9(LinkedList<Pet> list)

{

//9 - выполнение методов всех объектов, поддерживающих IFlying

foreach(Pet p in list)

{

if (p is IFlying)

{

((IFlying)p).Fly();

}

}

Console.ReadKey();

}

void method8(LinkedList<Pet> list)

{

//8 - реверс коллекции

Pet[] l = new Pet[list.Count];

var current = list.Last;

int i= 0;

while (current != null)

{

l[i]=current.Value;

current = current.Previous;

i++;

}

list.Clear();

for(int j=0; j < l.Length; j++)

{

list.AddLast(l[j]);

}

Console.WriteLine("Реверс завершен");

Console.ReadKey();

}

void method7(LinkedList<Pet> list)

{

//7 - удаление элемента в конце списка

list.RemoveLast();

Console.WriteLine("Элемент успешно удален");

Console.ReadKey();

}

void method6(LinkedList<Pet> list)

{

//6 - удаление элемента в начале списка

list.RemoveFirst();

Console.WriteLine("Элемент успешно удален");

Console.ReadKey();

}

void method5(LinkedList<Pet> list)

{

//5 - нахождение элемента с конца коллекции

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

var currentNode=list.Last;

while (currentNode != null)

{

if(currentNode.Value.Equals(pet))

{

Console.WriteLine($"Элемент найден: {currentNode.Value}");

Console.ReadKey();

return;

}

currentNode = currentNode.Previous;

}

Console.WriteLine("Элемент не найден");

Console.ReadKey();

}

void method4(LinkedList<Pet> list)

{

//4-нахождение элемента с начала коллекции

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

foreach (Pet p in list)

{

if(p.Equals(pet))

{

Console.WriteLine($"Элемент найден: {p}");

Console.ReadKey();

return;

}

}

Console.WriteLine("Элемент не найден");

Console.ReadKey();

}

void method3(LinkedList<Pet> list)

{

//3 - добавление элемента в начало списка

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name = Console.ReadLine();

Console.Write("age: ");

otvet = Console.ReadLine();

if (!Int32.TryParse(otvet, out age) || age < 0 || age > 30) return;

switch (n)

{

case 1: list.AddFirst(new Pet(name, age)); break;

case 2: list.AddFirst(new Bird(name, age)); break;

}

Console.WriteLine("Элемент добавлен в начало списка");

Console.ReadKey();

}

void method2(LinkedList<Pet> list)

{

//2 - доб-е эл-та в сп-к в конец списка

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name = Console.ReadLine();

Console.Write("age: ");

otvet = Console.ReadLine();

if (!Int32.TryParse(otvet, out age) || age < 0 || age > 30) return;

switch (n)

{

case 1: list.AddLast( new Pet(name, age)); break;

case 2: list.AddLast( new Bird(name, age)); break;

}

Console.WriteLine("Элемент добавлен в конец списка");

Console.ReadKey();

}

void method1(LinkedList<Pet> list)

{

//1-просмотр коллекции

int i = 0;

foreach (Pet p in list)

{

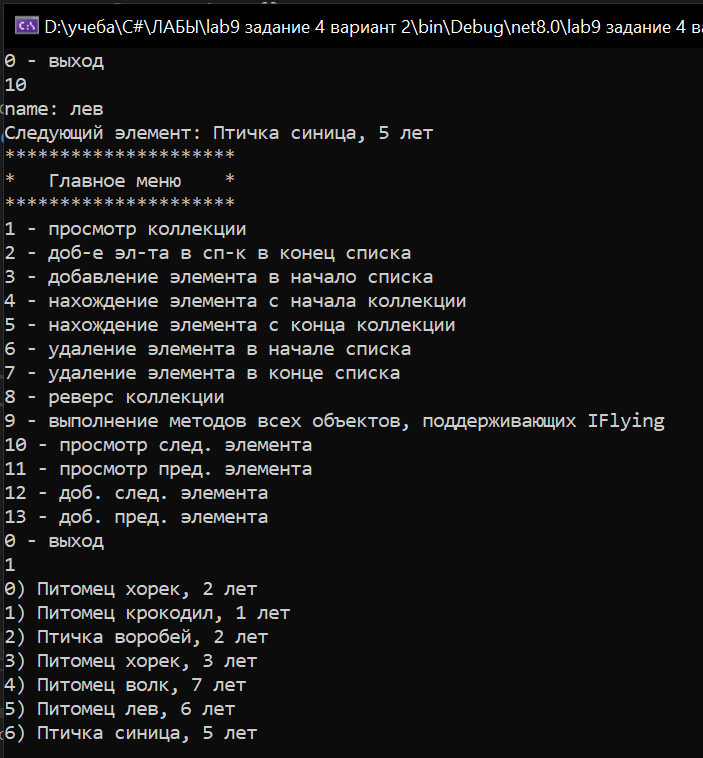
Console.WriteLine($"{i}) {p}");

i++;

}

Console.ReadKey();

}



Console.WriteLine("Лабораторная работа 9. Задание 4. Вариант 3");

Queue<Pet> queue = new Queue<Pet>();

queue.Enqueue(new Pet("хорек", 2));

queue.Enqueue(new Pet("крокодил", 1));

queue.Enqueue(new Bird("воробей", 2));

queue.Enqueue(new Pet("хорек", 3));

queue.Enqueue(new Pet("волк", 7));

queue.Enqueue(new Pet("лев", 6));

queue.Enqueue(new Bird("синица", 5));

while (true)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* Главное меню \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("1 - просмотр коллекции");

Console.WriteLine("2 - добавление элемента в конец очереди");

Console.WriteLine("3 - очищение очереди");

Console.WriteLine("4 - пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в очереди");

Console.WriteLine("5 - извлекает и возвращает первый эл. в очереди");

Console.WriteLine("6 - посмотреть первый эл. в очереди");

Console.WriteLine("7 - выполнение методов всех объектов, поддерживающих IFlying");

Console.WriteLine("0 - выход");

int otvet = Convert.ToInt32(Console.ReadLine());

switch (otvet)

{

case 1: method1(queue); break;

case 2: method2(queue); break;

case 3: method3(queue); break;

case 4: method4(queue); break;

case 5: method5(queue); break;

case 6: method6(queue); break;

case 7: method7(queue); break;

default: return;

}

}

void method7(Queue<Pet> queue)

{

//7 - выполнение методов всех объектов, поддерживающих IFlying

foreach (var pair in queue)

{

if (pair is IFlying)

{

((IFlying)pair).Fly();

}

}

Console.ReadKey();

}

void method6(Queue<Pet> queue)

{

//6 - извлекает первый эл. в очереди

Pet pet = queue.Peek();

Console.WriteLine("Очередь: ");

int i = 0;

foreach (var p in queue)

{

Console.WriteLine($"{i}) {p.ToString()}");

i++;

}

Console.WriteLine($"Первый элемент: {pet}");

Console.ReadKey();

}

void method5(Queue<Pet> queue)

{

//5 - извлекает и возвращает первый эл. в очереди

Pet pet = queue.Dequeue();

Console.WriteLine("Новая очередь: ");

int i = 0;

foreach(var p in queue)

{

Console.WriteLine($"{i}) {p.ToString()}");

i++;

}

Console.WriteLine($"Удаленный элемент: {pet}");

Console.ReadKey();

}

void method4(Queue<Pet> queue)

{

//пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в очереди

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet =new Pet(name);

Console.WriteLine(queue.Contains(pet) ? true : false);

foreach (var pair in queue)

{

if (pair.Name == pet.Name)

{

Console.WriteLine($"Искомый элемент: {pair}");

}

}

Console.ReadKey();

}

void method3(Queue<Pet> queue)

{

//3 - очищение очереди

queue.Clear();

Console.WriteLine("Очередь очищена");

Console.ReadKey();

}

void method2(Queue<Pet> queue)

{

//2 - добавление элемента в конец очереди

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name = Console.ReadLine();

Console.Write("age: ");

otvet = Console.ReadLine();

if (!Int32.TryParse(otvet, out age) || age < 0 || age > 30) return;

switch (n)

{

case 1: queue.Enqueue(new Pet(name, age)); break;

case 2: queue.Enqueue(new Bird(name, age)); break;

}

Console.WriteLine("Элемент добавлен");

Console.ReadKey();

}

void method1(Queue<Pet> queue)

{

//1-просмотр коллекции

int i = 0;

foreach (Pet p in queue)

{

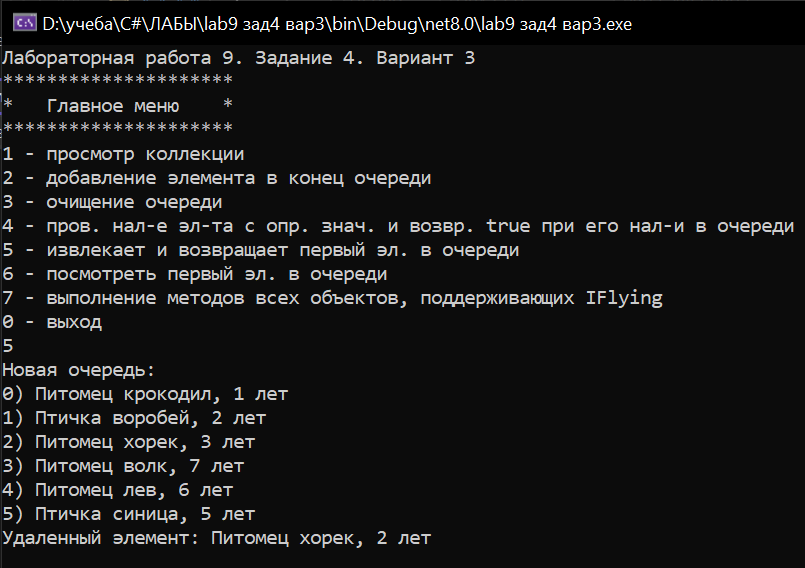
Console.WriteLine($"{i}) {p}");

i++;

}

Console.ReadKey();

}



Console.WriteLine("Лабораторная работа 9. Задание 4. вариант 4");

Stack<Pet> pets = new Stack<Pet>();

pets.Push(new Pet("хорек", 2));

pets.Push(new Pet("крокодил", 1));

pets.Push(new Bird("воробей", 2));

pets.Push(new Pet("хорек", 3));

pets.Push(new Pet("волк", 7));

pets.Push(new Pet("лев", 6));

pets.Push(new Bird("синица", 5));

while (true)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* Главное меню \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("1 - просмотр стека");

Console.WriteLine("2 - добавление элемента в стек в верхушку стека");

Console.WriteLine("3 - очищение стек");

Console.WriteLine("4 - пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в стеке");

Console.WriteLine("5 - извлекает и возвращает первый эл. в стеке");

Console.WriteLine("6 - посмотреть первый эл. в стеке");

Console.WriteLine("7 - выполнение методов всех объектов, поддерживающих IFlying");

Console.WriteLine("0 - выход");

int otvet = Convert.ToInt32(Console.ReadLine());

switch (otvet)

{

case 1: method1(pets); break;

case 2: method2(pets); break;

case 3: method3(pets); break;

case 4: method4(pets); break;

case 5: method5(pets); break;

case 6: method6(pets); break;

case 7: method7(pets); break;

default: return;

}

}

void method7(Stack<Pet> pets)

{

//7 - выполнение методов всех объектов, поддерживающих IFlying

foreach (var pair in pets)

{

if (pair is IFlying)

{

((IFlying)pair).Fly();

}

}

Console.ReadKey();

}

void method6(Stack<Pet> pets)

{

//6 - извлекает первый эл. в очереди

Pet pet = pets.Peek();

Console.WriteLine("Стек: ");

int i = 0;

foreach (var p in pets)

{

Console.WriteLine($"{i}) {p.ToString()}");

i++;

}

Console.WriteLine($"Первый элемент: {pet}");

Console.ReadKey();

}

void method5(Stack<Pet> pets)

{

//5 - извлекает и возвращает первый эл. из стека

Pet pet = pets.Pop();

Console.WriteLine("Новый стек: ");

int i = 0;

foreach (var p in pets)

{

Console.WriteLine($"{i}) {p.ToString()}");

i++;

}

Console.WriteLine($"Удаленный элемент: {pet}");

Console.ReadKey();

}

void method4(Stack<Pet> pets)

{

//пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в стеке

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

Console.WriteLine(pets.Contains(pet) ? true : false);

foreach (var pair in pets)

{

if (pair.Name == pet.Name)

{

Console.WriteLine($"Искомый элемент: {pair}");

}

}

Console.ReadKey();

}

void method3(Stack<Pet> pets)

{

//3 - очищение стека

pets.Clear();

Console.WriteLine("Стек очищен");

Console.ReadKey();

}

void method2(Stack<Pet> pets)

{

//2 - добавление элемента в стек в верхушку стека

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name = Console.ReadLine();

Console.Write("age: ");

otvet = Console.ReadLine();

if (!Int32.TryParse(otvet, out age) || age < 0 || age > 30) return;

switch (n)

{

case 1: pets.Push(new Pet(name, age)); break;

case 2: pets.Push(new Bird(name, age)); break;

}

Console.WriteLine("Элемент добавлен");

Console.ReadKey();

}

void method1(Stack<Pet> pets)

{

//1-просмотр стека

int i = 0;

foreach (Pet p in pets)

{

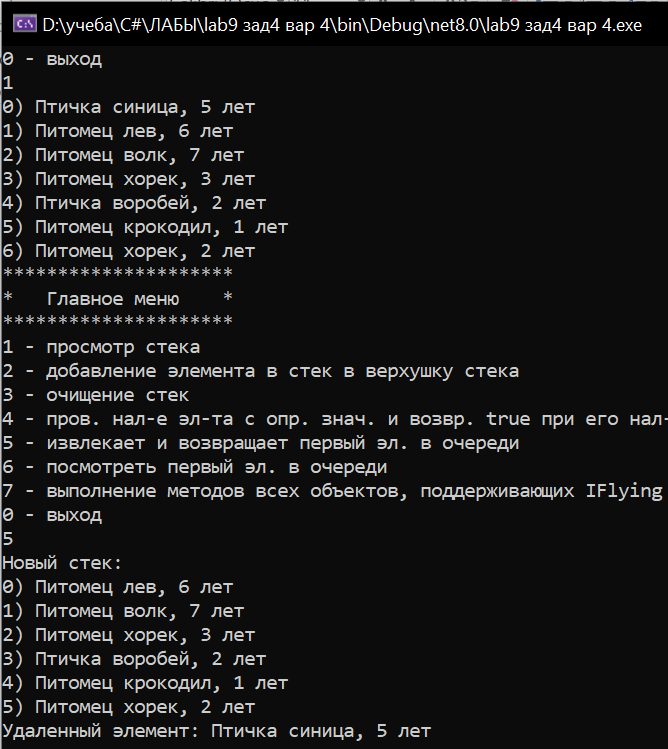
Console.WriteLine($"{i}) {p}");

i++;

}

Console.ReadKey();

}



Console.WriteLine("Лабораторная работа 9. Задание 4. Вариант 5");

HashSet<Pet> pets = new HashSet<Pet>();

pets.Add(new Pet("хорек", 2));

pets.Add(new Pet("крокодил", 1));

pets.Add(new Bird("воробей", 2));

pets.Add(new Pet("хорек", 3));

pets.Add(new Pet("волк", 7));

pets.Add(new Pet("лев", 6));

pets.Add(new Bird("синица", 5));

HashSet<Pet> pets1 = new HashSet<Pet>();

pets1.Add(new Pet("хорек1", 2));

pets1.Add(new Pet("крокодил", 1));

pets1.Add(new Bird("воробей1", 2));

pets1.Add(new Pet("хорек", 3));

pets1.Add(new Pet("волк", 7));

pets1.Add(new Pet("лев1", 6));

pets1.Add(new Bird("синица", 5));

while (true)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* Главное меню \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("1 - просмотр множества");

Console.WriteLine("2 - добавление элемента в множество");

Console.WriteLine("3 - очищение множества");

Console.WriteLine("4 - пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в множестве");

Console.WriteLine("5 - удаляет элемент в множестве");

Console.WriteLine("6 - объединить множества");

Console.WriteLine("7 - выполнение методов всех объектов, поддерживающих IFlying");

Console.WriteLine("0 - выход");

int otvet = Convert.ToInt32(Console.ReadLine());

switch (otvet)

{

case 1: method1(pets); break;

case 2: method2(pets); break;

case 3: method3(pets); break;

case 4: method4(pets); break;

case 5: method5(pets); break;

case 6: method6(pets, pets1); break;

case 7: method7(pets); break;

default: return;

}

}

void method7(HashSet<Pet> pets)

{

//7 - выполнение методов всех объектов, поддерживающих IFlying

foreach (var pair in pets)

{

if (pair is IFlying)

{

((IFlying)pair).Fly();

}

}

Console.ReadKey();

}

void method6(HashSet<Pet> pets, HashSet<Pet> pets1)

{

//6 - объединить множества

pets.UnionWith(pets1);

int i = 0;

foreach (var p in pets)

{

Console.WriteLine($"{i}) {p.ToString()}");

i++;

}

Console.ReadKey();

}

void method5(HashSet<Pet> pets)

{

//5 - удаляет элемент из множества

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

foreach (var p in pets)

{

if(p.Name == pet.Name)

{

Console.WriteLine(pets.Remove(p)?"Элемент успешно удален":"Элемент не нйден");

}

}

Console.ReadKey();

}

void method4(HashSet<Pet> pets)

{

//пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в множестве

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

Console.WriteLine(pets.Contains(pet) ? true : false);

foreach (var pair in pets)

{

if (pair.Name == pet.Name)

{

Console.WriteLine($"Искомый элемент: {pair}");

}

}

Console.ReadKey();

}

void method3(HashSet<Pet> pets)

{

//3 - очищение множества

pets.Clear();

Console.WriteLine("Множество очищено");

Console.ReadKey();

}

void method2(HashSet<Pet> pets)

{

//2 - добавление элемента в множество

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name = Console.ReadLine();

Console.Write("age: ");

otvet = Console.ReadLine();

if (!Int32.TryParse(otvet, out age) || age < 0 || age > 30) return;

switch (n)

{

case 1: pets.Add(new Pet(name, age)); break;

case 2: pets.Add(new Bird(name, age)); break;

}

Console.WriteLine("Элемент добавлен");

Console.ReadKey();

}

void method1(HashSet<Pet> pets)

{

//1-просмотр множества

int i = 0;

foreach (Pet p in pets)

{

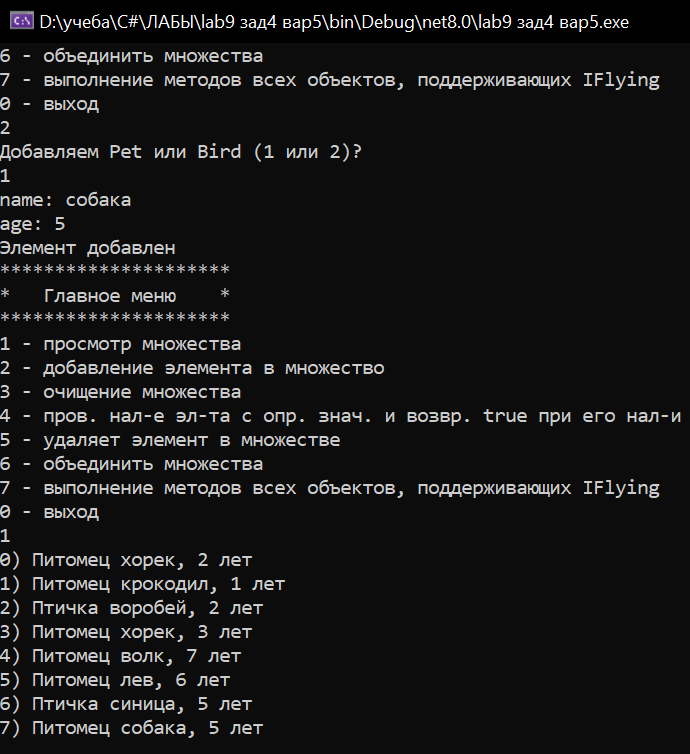
Console.WriteLine($"{i}) {p}");

i++;

}

Console.ReadKey();

}



Console.WriteLine("Лабораторная работа 9. Задание 4. Вариант 6");

List<Pet> list = new List<Pet>();

list.Add(new Pet("хорек", 2));

list.Add(new Pet("крокодил", 1));

list.Add(new Bird("воробей", 2));

list.Add(new Pet("хорек", 3));

list.Add(new Pet("волк", 7));

list.Add(new Pet("лев", 6));

list.Add(new Bird("синица", 5));

SortedList<int,Pet> pets= new SortedList<int,Pet>();

int i = 0;

foreach(Pet pet in list)

{

pets.Add(i,pet);

i++;

}

while (true)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* Главное меню \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("1 - просмотр отсортированной коллекции");

Console.WriteLine("2 - добавление элемента в отсортированную коллекцию");

Console.WriteLine("3 - очищение отсортированной коллекции");

Console.WriteLine("4 - пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в отсортированной коллекции");

Console.WriteLine("5 - удаляет элемент в отсортированной коллекции");

Console.WriteLine("6 - получить значение по индексу ");

Console.WriteLine("7 - выполнение методов всех объектов, поддерживающих IFlying");

Console.WriteLine("0 - выход");

int otvet = Convert.ToInt32(Console.ReadLine());

switch (otvet)

{

case 1: method1(pets); break;

case 2: method2(pets); break;

case 3: method3(pets); break;

case 4: method4(pets); break;

case 5: method5(pets); break;

case 6: method6(pets); break;

case 7: method7(pets); break;

default: return;

}

}

void method7(SortedList<int, Pet> pets)

{

//7 - выполнение методов всех объектов, поддерживающих IFlying

foreach (var pair in pets)

{

if (pair.Value is IFlying)

{

((IFlying)pair.Value).Fly();

}

}

Console.ReadKey();

}

void method6(SortedList<int, Pet> pets)

{

//6 - получить значение по индексу

Console.Write("индекс: ");

string name = Console.ReadLine();

int index;

if (!Int32.TryParse(name, out index) || index < 0 || index >= pets.Count) return;

Pet pet=pets.GetValueAtIndex(index);

Console.WriteLine($"Искомый элемент: {pet}");

Console.ReadKey();

}

void method5(SortedList<int, Pet> pets)

{

//5 - удаляет элемент из коллекции

Console.Write("key: ");

string name = Console.ReadLine();

int index;

if (!Int32.TryParse(name, out index) || index < 0 || index >= pets.Count) return;

foreach (var p in pets)

{

if (p.Key == index)

{

Console.WriteLine(pets.Remove(index) ? "Элемент успешно удален" : "Элемент не найден");

}

}

Console.ReadKey();

}

void method4(SortedList<int, Pet> pets)

{

//пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в множестве

Console.Write("key: ");

string name = Console.ReadLine();

int index;

if (!Int32.TryParse(name, out index) || index < 0 || index >= pets.Count) return;

Console.WriteLine(pets.ContainsKey(index) ? true : false);

foreach (var pair in pets)

{

if (pair.Key == index)

{

Console.WriteLine($"Key: {pair.Key} Искомый элемент: {pair.Value}");

}

}

Console.ReadKey();

}

void method3(SortedList<int, Pet> pets)

{

//3 - очищение отсортированной коллекции

pets.Clear();

Console.WriteLine("Коллекция очищена");

Console.ReadKey();

}

void method2(SortedList<int, Pet> pets)

{

//2 - добавление элемента в отсортированную коллекцию

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name = Console.ReadLine();

Console.Write("age: ");

otvet = Console.ReadLine();

if (!Int32.TryParse(otvet, out age) || age < 0 || age > 30) return;

switch (n)

{

case 1: pets.Add(pets.Count,new Pet(name, age)); break;

case 2: pets.Add(pets.Count, new Bird(name, age)); break;

}

Console.WriteLine("Элемент добавлен");

Console.ReadKey();

}

void method1(SortedList<int, Pet> pets)

{

//1-просмотр отсортированной коллекции

foreach ( KeyValuePair<int,Pet> p in pets)

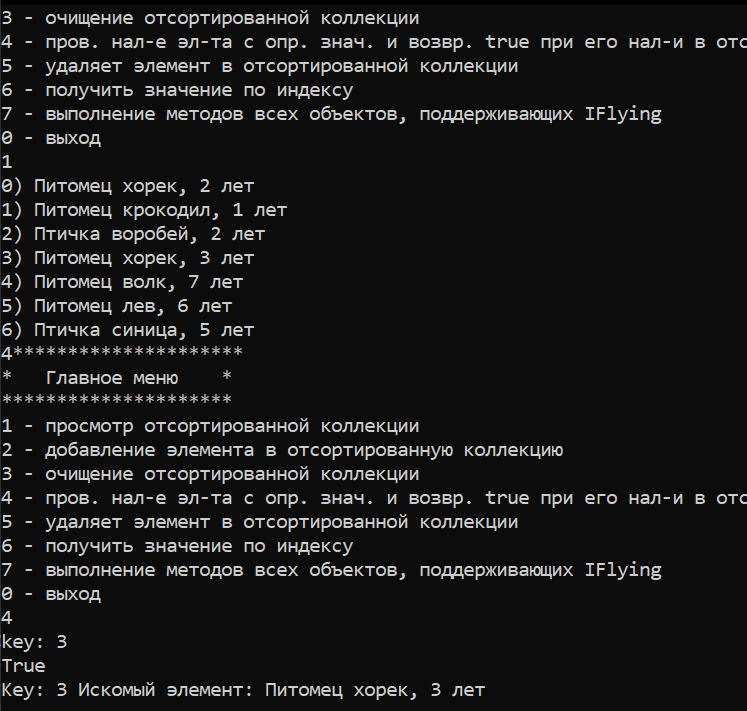
{

Console.WriteLine($"{p.Key}) {p.Value}");

}

Console.ReadKey();

}



Console.WriteLine("Лабораторная работа 9. Задание 4. Вариант 7");

SortedSet<Pet> pets = new SortedSet<Pet>();

pets.Add(new Pet("хорек", 2));

pets.Add(new Pet("крокодил", 1));

pets.Add(new Bird("воробей", 2));

pets.Add(new Pet("хорек", 3));

pets.Add(new Pet("волк", 7));

pets.Add(new Pet("лев", 6));

pets.Add(new Bird("синица", 5));

SortedSet<Pet> pets1 = new SortedSet<Pet>();

pets1.Add(new Pet("хорек1", 2));

pets1.Add(new Pet("крокодил", 1));

pets1.Add(new Bird("воробей1", 2));

pets1.Add(new Pet("хорек", 3));

pets1.Add(new Pet("волк", 7));

pets1.Add(new Pet("лев1", 6));

pets1.Add(new Bird("синица", 5));

while (true)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* Главное меню \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("1 - просмотр множества");

Console.WriteLine("2 - добавление элемента в множество");

Console.WriteLine("3 - очищение множества");

Console.WriteLine("4 - пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в множестве");

Console.WriteLine("5 - удаляет элемент в множестве");

Console.WriteLine("6 - объединить множества");

Console.WriteLine("7 - выполнение методов всех объектов, поддерживающих IFlying");

Console.WriteLine("0 - выход");

int otvet = Convert.ToInt32(Console.ReadLine());

switch (otvet)

{

case 1: method1(pets); break;

case 2: method2(pets); break;

case 3: method3(pets); break;

case 4: method4(pets); break;

case 5: method5(pets); break;

case 6: method6(pets, pets1); break;

case 7: method7(pets); break;

default: return;

}

}

void method7(SortedSet<Pet> pets)

{

//7 - выполнение методов всех объектов, поддерживающих IFlying

foreach (var pair in pets)

{

if (pair is IFlying)

{

((IFlying)pair).Fly();

}

}

Console.ReadKey();

}

void method6(SortedSet<Pet> pets, SortedSet<Pet> pets1)

{

//6 - объединить множества

pets.UnionWith(pets1);

int i = 0;

foreach (var p in pets)

{

Console.WriteLine($"{i}) {p.ToString()}");

i++;

}

Console.ReadKey();

}

void method5(SortedSet<Pet> pets)

{

//5 - удаляет элемент из множества

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

foreach (var p in pets)

{

if (p.Name == pet.Name)

{

Console.WriteLine(pets.Remove(p) ? "Элемент успешно удален" : "Элемент не нйден");

}

}

Console.ReadKey();

}

void method4(SortedSet<Pet> pets)

{

//пров. нал-е эл-та с опр. знач. и возвр. true при его нал-и в множестве

Console.Write("name: ");

string name = Console.ReadLine();

Pet pet = new Pet(name);

Console.WriteLine(pets.Contains(pet) ? true : false);

foreach (var pair in pets)

{

if (pair.Name == pet.Name)

{

Console.WriteLine($"Искомый элемент: {pair}");

}

}

Console.ReadKey();

}

void method3(SortedSet<Pet> pets)

{

//3 - очищение множества

pets.Clear();

Console.WriteLine("Множество очищено");

Console.ReadKey();

}

void method2(SortedSet<Pet> pets)

{

//2 - добавление элемента в множество

Console.WriteLine("Добавляем Pet или Bird (1 или 2)?");

string otvet = Console.ReadLine();

int n, age;

if (!Int32.TryParse(otvet, out n) || n < 1 || n > 2) return;

Console.Write("name: ");

string name = Console.ReadLine();

Console.Write("age: ");

otvet = Console.ReadLine();

if (!Int32.TryParse(otvet, out age) || age < 0 || age > 30) return;

switch (n)

{

case 1: pets.Add(new Pet(name, age)); break;

case 2: pets.Add(new Bird(name, age)); break;

}

Console.WriteLine("Элемент добавлен");

Console.ReadKey();

}

void method1(SortedSet<Pet> pets)

{

//1-просмотр множества

int i = 0;

foreach (Pet p in pets)

{

Console.WriteLine($"{i}) {p}");

i++;

}

Console.ReadKey();

}

