Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5 «Информационные системы и программная инженерия»

**Лабораторная работа №2**по дисциплине «Программирование на языке высокого уровня»  
по теме «Разработка программы для работы со встроенными классами, обеспечивающими работу с файлами, с использованием потоков»

Выполнил:  
Студент Альков В. С.  
Группа И407Б  
Вариант 2  
  
Преподаватель:  
Кимсанбаев К. А.

Санкт-Петербург  
2021 г.

Задача:

Написать программу для работы с данными, представленными в виде набора структур, содержимое которых соответствует индивидуальному варианту. Данные считываются из файла и заносятся в указанную в варианте коллекцию из числа стандартных. При старте программы пользователю предлагается ввести имя файла. Если файл не существует, создаётся новый. После этого пользователю предлагается меню для работы с данными. Обязательные пункты меню: отображение содержимого коллекции, добавление нового элемента, удаления элемента с указанным индексом, корректировка элемента, работа с коллекцией, дополнительные пункты указаны в варианте. После работы необходимо сохранять содержимое коллекции в указанные ранее пользователем файл. Вывод данных на экран должен выполняться в формате таблицы.

Спортсмены. Коллекция - двусвязный список. Поля данных структуры: фамилия, пол, вид спорта, год рождения, рост. Дополнительные пункты меню: найти самого высокого спортсмена, занимающегося плаванием, среди мужчин. Вывести сведения о спортсменках, выступающих в юниорском разряде (14-17лет).

# Вспомогательные функции и классы

## Функция поиска элемента в списке по имени.

Входные данные: список, имя.

Выходные данные: найденный элемент или null.

public static LinkedListNode<Sportsman>?

FindNodeByName(LinkedList<Sportsman> list, string name)

{

var f = list.FirstOrDefault(delegate (Sportsman o)

{

if (o.surname.ToLower().Equals(name.ToLower()))

return true;

return false;

});

return list.Find(f);

}

## Функция поиска элемента в списке по индексу.

Входные данные: список, индекс.

Выходные данные: найденный элемент или null.

public static LinkedListNode<Sportsman>?

FindNodeByIndex(LinkedList<Sportsman> list, int index)

{

var f = list.ElementAt(index);

return list.Find(f);

}

## Функция поиска спортсменок от 14 до 17 лет.

Входные данные: список.

Выходные данные: список спортсменок.

public static LinkedList<Sportsman> FindWomen(LinkedList<Sportsman> list)

{

return new LinkedList<Sportsman>(list.Where(delegate (Sportsman o)

{

if (2021 - o.year >= 14 && 2021 - o.year <= 17 && o.sex.ToLower()[0].Equals('ж'))

return true;

return false;

}));

}

## Функция поиска самого высокого спортсмена, занимающегося плаванием среди мужчин в списке.

Входные данные: список.

Выходные данные: список с объектом или пустой список.

public static LinkedList<Sportsman> FindHighest(LinkedList<Sportsman> list)

{

var res = new LinkedList<Sportsman>();

Sportsman? entry = null;

int height = 0;

foreach(var i in list)

{

if(i.sport.ToLower().Equals("плавание")

&& i.sex.ToLower()[0].Equals('м') && i.height > height)

{

height = i.height;

entry = i;

}

}

if (entry != null)

res.AddLast((Sportsman)entry);

return res;

}

## Функция печати списка.

Входные данные: список.

Выходные данные: нет.

public static void PrintEntriesList(LinkedList<Sportsman> list)

{

int j = 0;

Console.WriteLine("{0,6}{1,15}{2,20}{3,15}{4,18}{5,11}"

,"Индекс", "Фамилия", "Вид спорта", "Пол", "Год рождения", "Рост");

foreach (var i in list)

Console.WriteLine(String.Format("{0,6}{1,15}{2,20}{3,15}{4,18}{5,11}",

++j, i.surname.CutString(10), i.sport.CutString(15), i.sex.CutString(10),

i.year.ToString().CutString(6), i.height.ToString().CutString(5)));

}

## Функция чтения данных из файла.

Входные данные: название файла.

Выходные данные: список прочитанных данных.

public static LinkedList<Sportsman> ReadFile(string filename)

{

using var file = new FileStream(filename, FileMode.OpenOrCreate);

using var reader = new StreamReader(file);

try

{

return JsonSerializer.Deserialize<LinkedList<Sportsman>>

(reader.ReadToEnd(), new JsonSerializerOptions{IncludeFields = true});

}

catch(JsonException)

{

Console.Write("Файл пуст, либо некоректен. Нажмите Enter, чтобы прододжить...");

Console.ReadLine();

return new LinkedList<Sportsman>();

}

}

## Функция записи списка в файл.

Входные данные: имя файла, список.

Выходные данные: нет.

public static void WriteFile(string filename, LinkedList<Sportsman> data)

{

using var file = new FileStream(filename, FileMode.Create);

using var writer = new StreamWriter(file);

var a = JsonSerializer.Serialize(data, new JsonSerializerOptions { IncludeFields = true });

writer.Write(a);

}

## Функция чтения строки с выводом заданного сообщения.

Входные данные: сообщение.

Выходные данные: прочитанная строка.

public static string ReadString(string message = "")

{

Console.Clear();

Console.Write(message);

return Console.ReadLine();

}

## Функция чтения int из диапазона, с выводом сообщения.

Входные данные: начало диапазона, конец диапазона, сообщение.

Выходные данные: число.

public static int ReadInt(int p1, int p2, string message = "")

{

int input;

bool success;

do

{

Console.Clear();

Console.Write(message);

success = int.TryParse(Console.ReadLine(), out input);

} while (!success || input < p1 || input > p2);

return input;

}

## Класс пункт меню.

[Flags]

public enum ItemType //тип пункта

{

None = 0,

Default = 1, //отображает детей для выбора

Move = 2, //пункт перехода в другой пункт

Exit = 4, //при переходе будет осуществлен выход

}

[Flags]

public enum ItemFlags

{

None = 0,

Action = 1, //сигнал о необходимости выполнить переданное действие

ClearScreen = 2, //флаг очистки экрана, после выполнения действий

Pause = 4, //флаг паузы, посредством чтения строки, после выполнения действий

}

class Item

{

ItemType type;

ItemFlags flags;

List<Item> items; //список подпунктов

Item? parent; //родитель пункта

Item? moveItem; //элемент для перехода

string? text; //название пункта

Action? action; //действие

public Item() : this(null, ItemType.Default) { }

public Item(string? text, ItemType type, ItemFlags flags = ItemFlags.None,

Action? action = null, Item? moveItem = null)

{

this.items = new List<Item>();

this.parent = null;

this.text = text;

this.type = type;

this.flags = flags;

this.moveItem = moveItem;

this.action = action;

}

public Item Add(Item newItem)

{

newItem.parent = this;

items.Add(newItem);

return newItem;

}

public Item? Update()

{

Item? tmp = this;

try

{

tmp.UpdateFlags(); //выполнение флагов

}

catch (ReturnToParentException) //обработка исключения

{

tmp = tmp.parent;

};

if (tmp != null) //обработка перехода в другие пункты

{

if (tmp.type.HasFlag(ItemType.Exit))

return tmp = null;

if (tmp.type.HasFlag(ItemType.Move))

tmp = tmp.moveItem;

if (tmp != null && tmp.type.HasFlag(ItemType.Default))

{

if (tmp.items.Count == 0)

tmp = parent;

else //выбор пользователя в меню

tmp=tmp.items[ReadInt(1,tmp.items.Count,tmp.ItemsTextToString())-1];

}

}

return tmp;

}

public void UpdateFlags()

{

if (flags.HasFlag(ItemFlags.ClearScreen))

Console.Clear();

if (flags.HasFlag(ItemFlags.Action) && action != null)

action();

if (flags.HasFlag(ItemFlags.Pause))

{

Console.Write("Нажмите Enter чтобы продожить..");

Console.ReadLine();

}

}

public string ItemsTextToString()

{

string print = "";

foreach (var i in this.items)

print += i + "\n";

return print;

}

public override string ToString() => text ?? ""; //если text == null, то text = ""

}

## Класс меню.

class Menu : Item

{

Item? current;

public Menu() { }

public void Run() //запуск меню

{

current = this;

while (current != null)

current = current.Update();

}

public void SetCurrent(Item current) => this.current = current;

}

## Класс исключения по переходу в родителя.

public class ReturnToParentException : Exception {}

## Класс меню помощник в работе со строками

public static class StringHelper

{

//функция обрезания строки до переданного количества, учитывая длину строки

public static string CutString(this string str, int lenght)

{

return str.Length > lenght ? str.Substring(0, lenght - 3) + "..." : str;

}

}

## Структура спортсмен

public struct Sportsman

{

//атрибут включения считывания поля для Json

[JsonInclude]

public string surname, sport, sex;

[JsonInclude]

public int year, height;

public Sportsman(string surname, string sport, string sex, int year, int height)

{

this.surname = surname;

this.sport = sport;

this.sex = sex;

this.year = year;

this.height = height;

}

public static Sportsman CreateInstanseFromConsole() //метод создания объекта

{

return new Sportsman(ReadString("Фамилия: "), ReadString("Вид Спорта: "),

ReadString("Пол: "), ReadInt(1900, 2021, "Год рождения: "),

ReadInt(0, 1000, "Рост: "));

}

public override string ToString()

{

return String.Format("{0}\n{1}\n{2}\n{3}\n{4}",

surname, sport, sex, year, height);

}

}

## Главная функция.

public static void Main()

{

LinkedListNode<Sportsman> objectNode = null;

Sportsman objectValue = new();

Console.Write("Введите имя файла: ");

string filename = Console.ReadLine();

LinkedList<Sportsman> list = ReadFile(filename);

var menu = new Menu();

var item1 = menu.Add(new Item("1. Добавить запись", ItemType.Default, ItemFlags.ClearScreen));

item1.Add(new Item("1. В начало", ItemType.Move, ItemFlags.Action | ItemFlags.ClearScreen,

delegate () { list.AddFirst(Sportsman.CreateInstanseFromConsole()); }, menu));

item1.Add(new Item("2. В конец", ItemType.Move, ItemFlags.Action | ItemFlags.ClearScreen,

delegate () { list.AddLast(Sportsman.CreateInstanseFromConsole()); }, menu));

item1.Add(new Item("3. В произвольное место", ItemType.Move,

ItemFlags.Action | ItemFlags.ClearScreen | ItemFlags.Pause, delegate ()

{

if (list.Count != 0)

{

Console.Write("Введите фамилию записи, после которой вставить: ");

var node = FindNodeByName(list, Console.ReadLine());

if (node != null)

{

list.AddAfter(node, Sportsman.CreateInstanseFromConsole());

Console.WriteLine("Запись добавлена");

}

else

{

Console.WriteLine("Запись не найдена");

}

}

else

Console.WriteLine("Коллекция пуста");

}, menu));

menu.Add(new Item("2. Вывести записи", ItemType.Default,

ItemFlags.Action | ItemFlags.ClearScreen | ItemFlags.Pause,

delegate () { PrintEntriesList(list); }));

menu.Add(new Item("3. Найти самого высокого спортсмена, занимающегося плаванием, среди мужчин",

ItemType.Default, ItemFlags.Action | ItemFlags.ClearScreen | ItemFlags.Pause,

delegate () { PrintEntriesList(FindHighest(list)); }));

menu.Add(new Item("4. Вывести сведения о спортсменках, выступающих в юниорском разряде (14 - 17лет)",

ItemType.Default, ItemFlags.Action | ItemFlags.ClearScreen | ItemFlags.Pause,

delegate () { PrintEntriesList(FindWomen(list)); }));

menu.Add(new Item("5. Отсортировать по имени", ItemType.Default,

ItemFlags.Action | ItemFlags.ClearScreen, delegate () {

list = new LinkedList<Sportsman>(list.OrderBy(a => a.surname)); }));

menu.Add(new Item("6. Удалить запись по фамилии", ItemType.Default,

ItemFlags.Action | ItemFlags.ClearScreen | ItemFlags.Pause, delegate () {

Console.Write("Введите фамилию: ");

var entry = FindNodeByName(list, Console.ReadLine());

if (entry != null)

{

Console.WriteLine("Запись удалена");

list.Remove(entry);

}

else

Console.WriteLine("Запись не найдена");

}));

menu.Add(new Item("7. Удалить запись по индеку", ItemType.Default,

ItemFlags.Action | ItemFlags.ClearScreen | ItemFlags.Pause,

delegate () {

if (list.Count > 0)

{

list.Remove(FindNodeByIndex(list,

ReadInt(1, list.Count, $"Введите индекс элемента ( от 1 до {list.Count} ): ") - 1));

Console.WriteLine("Запись удалена");

}

else

Console.WriteLine("Список пуст");

}));

var item2 = menu.Add(new Item("8. Изменить запись", ItemType.Default,

ItemFlags.Action | ItemFlags.ClearScreen,

delegate ()

{

if (objectNode == null)

{

Console.Write("Введите имя записи: ");

objectNode = FindNodeByName(list, Console.ReadLine());

if (objectNode == null)

{

Console.WriteLine("Запись не найдена. Нажмите Enter, чтобы продолжить...");

Console.ReadLine();

throw new ReturnToParentException();

}

objectValue = objectNode.Value;

}

}));

item2.Add(new Item("1. Фамилия", ItemType.Default, ItemFlags.Action | ItemFlags.ClearScreen,

delegate () { objectValue.surname = ReadString("Фамилия: "); }, menu));

item2.Add(new Item("2. Спорт", ItemType.Default, ItemFlags.Action | ItemFlags.ClearScreen,

delegate () { objectValue.sport = ReadString("Вид спорта: "); }, menu));

item2.Add(new Item("3. Пол", ItemType.Default, ItemFlags.Action | ItemFlags.ClearScreen,

delegate () { objectValue.sex = ReadString("Пол: "); }, menu));

item2.Add(new Item("4. Год рождения", ItemType.Default, ItemFlags.Action | ItemFlags.ClearScreen,

delegate () { objectValue.year = ReadInt(1900, 2021, "Год рождения: "); }, menu));

item2.Add(new Item("5. Рост", ItemType.Default, ItemFlags.Action | ItemFlags.ClearScreen,

delegate () { objectValue.height = ReadInt(0, 300, "Рост: "); }, menu));

item2.Add(new Item("6. Назад", ItemType.Move, ItemFlags.Action | ItemFlags.ClearScreen,

delegate () { objectNode.Value = objectValue; objectNode = null; }, menu));

menu.Add(new Item("9. Сохранить и выйти", ItemType.Exit, ItemFlags.Action,

delegate () { WriteFile(filename, list); }));

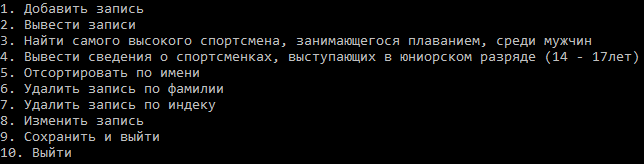
menu.Add(new Item("10. Выйти", ItemType.Exit, ItemFlags.None));

menu.Run();

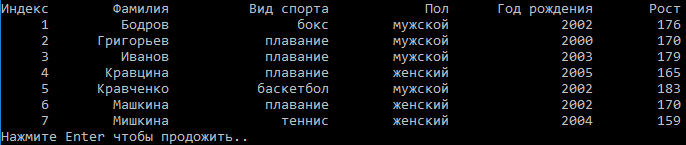
}

## Результат работы программы





### 2. Вывести записи



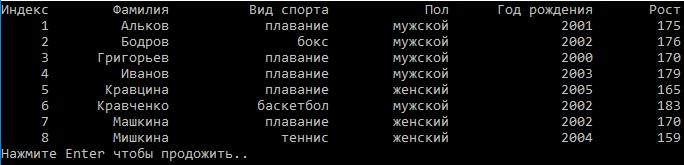
### 1. Добавить запись



### 1. В начало

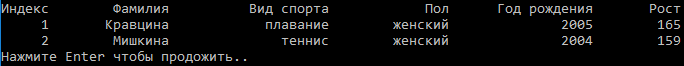
### 2. Вывести записи



### 3. Найти самого высокого спортсмена, занимающегося плаванием, среди мужчин



### 4. Вывести сведения о спортсменках, выступающих в юниорском разряде (14 - 17лет)



### 6. Удалить запись по фамилии

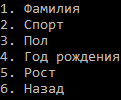


### 7. Удалить запись по индексу

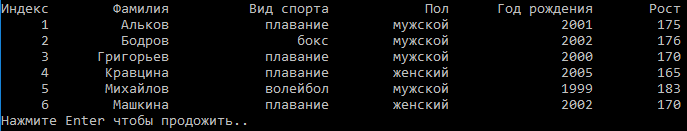
### 

### 8. Изменить запись

### 



### 2. Вывести записи



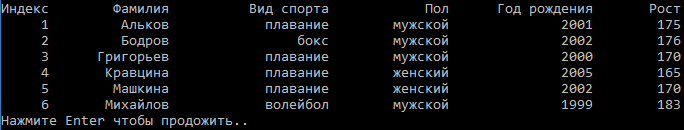
### 3. Найти самого высокого спортсмена, занимающегося плаванием, среди мужчин



### 4. Вывести сведения о спортсменках, выступающих в юниорском разряде (14 - 17лет)



### 2. Вывести записи



## Выводы

В этой практической работе были разработаны:

-структурный тип Спортсмен;

-функции для удобной работы с коллекцией типа Спортсмен:

-запись в файл коллекции;

-чтение из файла коллекции;

-удаления по индексу из коллекции;

-удаление по имени из коллекции;

-печать в табличном виде коллекции;

-поиск по коллекции одного элемента;

-поиск по коллекции нескольких элементов;

-получен опыт по работе с библиотекой JSON в сериализации и десериализации объектов;