Практическое занятие № 15

Тема практического занятия: Работа с коллекциями.

Цель практического занятия: овладение методами и алгоритмами работы с коллекциями для эффективного использования их для решения задач в программировании.

В результате выполнения данной работы обучающийся должен уметь:

1. Использовать коллекции для хранения и управления наборами данных
2. Создавать и инициализировать различные типы коллекций.
3. Выполнять операции добавления, удаления и изменения элементов в коллекциях.
4. Пользоваться встроенными методами и функциями для работы с коллекциями.
5. Использовать различные методы и алгоритмы для работы с коллекциями, включая сортировку и поиск элементов.
6. Применять концепции и принципы выбора подходящей коллекции для различных задач и требований.
7. Решать задачи и разрабатывать программы, используя знания и навыки работы с коллекциями.

знать: что

1. Основные принципы работы с коллекциями данных
2. Различные методы и алгоритмы для работы с коллекциями
3. Различия между различными типами коллекций и их особенностями в использовании.

Перечень оборудования, необходимого для выполнения задания:

- Автоматизированные рабочие места по количеству обучающихся (процессор Intel Core i7 или аналогичный, БП 700 Вт, 32 Гб ОЗУ, SSD 512 Гб, HDD 4 ТБ SATA 7200 rpm, RTX 3060 12GB);
- Автоматизированное рабочее место преподавателя (процессор Intel Core i7 или аналогичный, БП 700 Вт, 32 Гб ОЗУ, SSD 512 Гб, HDD 4 ТБ SATA 7200 rpm, RTX 3060 12GB);
- Монитор 34”, изогнутый, 3440x1440, 6 ms, 178°/178°, 300 cd/m2, 20M:1, HDMI, DP, USB, регулировка по высоте (1 монитор на одно рабочее место).

Программное обеспечение:

- Microsoft Visual Studio.

Общие теоретические сведения:

Коллекции - это структуры данных, предназначенные для хранения и управления группами элементов. Они позволяют удобно организовать и обрабатывать данные, предоставляя различные методы и функции для добавления, удаления, изменения и поиска элементов внутри коллекции.

Особенности обработки коллекций включают:

1. Гибкость: Коллекции позволяют хранить элементы разных типов и размеров. Это позволяет эффективно работать с данными различной природы.

2. Доступ к элементам: К элементам в коллекции можно обратиться по индексу, ключу или другому идентификатору. Это позволяет быстро получать нужные элементы для работы.

3. Методы и функции обработки: Коллекции обычно предоставляют набор методов и функций для выполнения таких операций, как сортировка, фильтрация, поиск, итерация и многое другое. Это упрощает обработку данных и позволяет реализовать различные алгоритмы и операции над коллекциями.

4. Управление размером и структурой: Коллекции позволяют динамически изменять размер и структуру данных, позволяя добавлять и удалять элементы при необходимости. Это удобно в случаях, когда нужно адаптироваться к изменяющимся данным или выполнять операции расширения или сжатия коллекции.

5. Итерация: Обычно коллекции имеют встроенную поддержку итерации, что позволяет эффективно обойти все элементы коллекции и выполнить необходимые операции над ними.

Особенности обработки коллекций могут различаться в зависимости от конкретных типов коллекций и используемых алгоритмов, но общая задача состоит в обеспечении удобной структуры для хранения, управления и обработки группы элементов данных.

Существует множество различных видов коллекций, предназначенных для хранения и управления различными типами данных. Ниже представлены некоторые из наиболее распространенных видов коллекций:

1. Списки: Коллекция, которая представляет последовательность элементов в определенном порядке. Разрешает дублирование элементов и обеспечивает доступ к элементам по индексу.

2. Массивы: Фиксированный размер коллекции, которая хранит элементы определенного типа. Обеспечивает доступ к элементам по индексу, но не позволяет динамическое изменение размера.

3. Множества: Коллекция, которая хранит уникальные элементы без определенного порядка. Не позволяет дублирование элементов.

4. Словари: Коллекция, которая представляет пары ключ-значение. Позволяет быстрый доступ к значениям по ключу.

5. Очереди: Коллекция, представляющая структуру данных «первым пришел, первым ушел» (FIFO). Элементы добавляются в конец очереди и удаляются из начала.

6. Стеки: Коллекция, представляющая структуру данных «последним пришел, первым ушел» (LIFO). Элементы добавляются и удаляются с одного конца стека.

7. Строки: Коллекция символов, используется для работы с текстовыми данными.

8. Другие специализированные коллекции: Такие как битовый массив, список на основе связанных элементов, отсортированный словарь и т.д.

Конкретный выбор типа коллекции зависит от требований и специфики задачи, которую нужно решить. Каждый вид коллекции имеет свои особенности использования и подходит для определенных сценариев работы с данными.

Задание:

1. В данном задании вам будет необходимо самим выбрать, какие коллекции будет целесообразно использовать.
2. Доработайте ранее разработанную библиотеку классов, добавив к классу «Достопримечательность» две координаты, характеризующие положение достопримечательности в городе.
3. Доработайте приложение, разработанное во время практического занятия №14, добавив возможность вводить координаты достопримечательности.
4. Добавьте в приложение возможность сформировать таблицу, отображающую расстояния между разными достопримечательностями. Таблица может выглядеть, например, так (приведен пример для четырех достопримечательностей):

	Музей	Ратуша	Театр	Парк
Музей	0	4	2	8
Ратуша	4	0	7	5
Театр	2	7	0	4
Парк	8	5	4	0

Сетка необязательна. Обратите внимание на то, что таблица симметрична.

5. Добавьте возможность ввести маршрут между различными достопримечательностями (необязательно включать все достопримечательности, порядок может быть произвольным, можно возвращаться к каким-то достопримечательностям, нельзя только посетить одну и ту же достопримечательность два раза подряд) и расчет расстояния при движении по этому маршруту. Если маршрут некорректен, выдайте соответствующее сообщение.

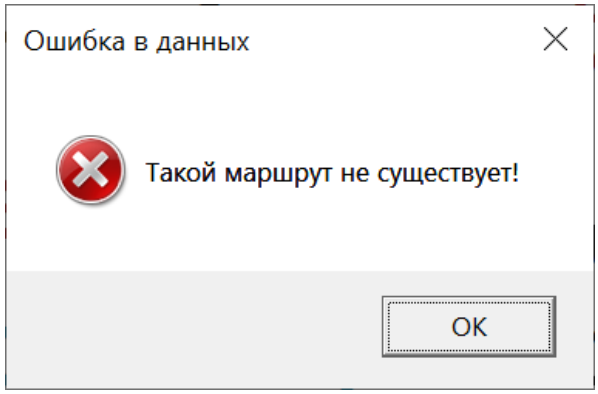
Указания по технике безопасности:

Инструкция по технике безопасности при работе в лаборатории, оборудованной компьютерной техникой.

Технология выполнения работы (*этапы, последовательность действий*):

1. Доработайте ранее созданный класс «Достопримечательность». Доработку производите в той же среде разработки, в которой вы разрабатывали класс.
2. Доработайте конструктор класса «Город», изменив инициализацию списка достопримечательностей с учетом новых полей.

3. Пересоберите библиотеку классов, убедитесь в успешности сборки. Чтобы пересобрать библиотеку, зайдите в пункт «Сборка», выберите действие «Пересобрать решение».
4. Доработайте интерфейс разработанного во время предыдущего практического занятия приложения. Доработку производите в той же среде разработки, в которой вы разрабатывали приложение.
5. Выберите тип коллекции для хранения таблицы с расстояниями. Выбор вы должны сделать самостоятельно. Возможные варианты: одномерный массив, двумерный массив, список, стек, очередь, словарь.
6. Реализуйте расчет расстояний и вывод таблицы.
7. Проверьте работоспособность этой части приложения.
8. Выберите тип коллекции для хранения информации о маршруте. Выбор вы должны сделать самостоятельно. Возможные варианты: одномерный массив, двумерный массив, список, стек, очередь, словарь.
9. Добавьте возможность вводить маршрут.
10. Реализуйте расчет длины маршрута.
11. Реализуйте обработку ситуации, когда маршрут некорректен. Например, может быть выдано сообщение:



12. Проверьте работоспособность этой части приложения.
    Ниже приведены примеры работы с различными типами коллекций.
    Работа с одномерным массивом:

```
static void Main(string[] args)
        {
            int[]a = new int[10000];
            Random rnd = new Random();
            for (int i = 0; i < 10000; i++)
                a[i] = rnd.Next(-1000, 1001);
            int m = a[0];
            int posMax = 0;
            for (int i = 0; i < a.Length; i++)
                if (a[i] > m)
                {
                    m = a[i];
                    posMax = i;
                }
            m = a[0];
            int posMin = 0;
            for (int i = 0; i < a.Length; i++)
                if (a[i] < m)
                {
                    m = a[i];
                    posMin = i;
                }
            a[posMin] = a[posMax];
            a[posMax] = m;
            foreach (int x in a)
                Console.WriteLine(x);
            Console.ReadKey();
        }
```

Работа с двумерным массивом:

```
static void Main(string[] args)
         {
             int[,] a = new int[10000,50];
             int[,] b = new int[10000,50];
             int[,] c = new int[10000, 50];
             Random rnd = new Random();
             for (int i = 0; i < 10000; i++)
                 for(int j = 0; i < 50;j++)
                     a[i, j] = rnd.Next(-1000, 1001);
             for (int i = 0; i < 10000; i++)
                 for (int j = 0; i < 50; j++)
                     b[i, j] = rnd.Next(-1000, 1001);
             for (int i = 0; i < 10000; i++)
                 for (int j = 0; i < 50; j++)
                     c[i, j] = a[i, j] + b[i, j];
             for (int i = 0; i < 10000; i++)
             {
                 for (int j = 0; i < 50; j++)
                     Console.Write(c[i, j] + "\t");
                 Console.WriteLine();
             }
             Console.ReadKey();
         }
```

Работа со списком:

```
static void Main(string[] args)
        {
            List<int> list = new List<int>();
            Random rnd = new Random();
            bool answer = false;
            for (int i = 0; i < 10000; i++)
                list.Add(rnd.Next(-100000, 100001));
            for (int i = 0; i < 10000; i++)
                for (int j = 0; j < 10000; j++)
                    if (list[i] == list[j] && i != j)
                        answer = true;
            Console.WriteLine(answer ? "Есть" : "Нет");
            Console.ReadKey();
        }
```

Работа со словарем:

```
var people = new Dictionary<int, string>()
            {
                [5] = "Tom",
                [6] = "Sam",
                [7] = "Bob",
            };
            // получаем элемент по ключу 6
            string sam = people[6];  // Sam
            Console.WriteLine(sam);  // Sam
                                     // переустанавливаем
значение по ключу 6
            people[6] = "Mike";
            Console.WriteLine(people[6]);  // Mike

            // добавляем новый элемент по ключу 22
            people[22] = "Eugene";
              Console.WriteLine(people[22]);  // Eugene
```

Требование к отчету:

1. Класс «Достопримечательность» содержит координаты достопримечательности.
2. Конструктор класса «Город» позволяет инициализировать список достопримечательностей с учетом координат.
3. Приложение позволяет ввести сведения о городе и достопримечательностях, включая координаты.

4. Приложение корректно строит таблицу с расстояниями между достопримечательностями.
5. Приложение корректно рассчитывает длину маршрута.
6. Приложение выдает сообщение, если маршрут некорректен.

Контрольные вопросы:

1. Что такое коллекции данных?
2. Какие основные типы коллекций существуют?
3. В чем различия между списками, множествами и словарями?
4. Как добавить элемент в список? Как удалить элемент из списка?
5. Что такое итерация по коллекции? Как реализовать итерацию?
6. Что такое ключ/значение в словаре? Как получить значение по ключу?
7. Какой тип коллекции лучше использовать для хранения уникальных элементов?
8. Какие методы доступны для сортировки элементов в коллекции?
9. Какой тип коллекции лучше использовать для выполнения операций "первым пришел, первым ушел" (FIFO)?
10. Какой тип коллекции лучше использовать для выполнения операций "последним пришел, первым ушел" (LIFO)?
11. Как выбрать подходящий тип коллекции для конкретной задачи?
12. Какие действия можно выполнить с коллекцией, помимо добавления и удаления элементов?
13. Как можно проверить наличие элемента в коллекции?

Основные и дополнительные источники, электронные ресурсы:

1. Подбельский, В. В. Язык C#. Базовый курс: учебное пособие / В. В. Подбельский. - 2-е изд., перераб. и доп. - Москва: Финансы и статистика, 2022. - 408 с. - ISBN 978-5-00184-079-4. - Текст: электронный. - URL: https://znanium.com/catalog/product/1913989
2. Гуриков, С. Р. Основы алгоритмизации и программирования на Python: учебное пособие / С.Р. Гуриков. — Москва: ИНФРА-М, 2023. — 343 с. — (Среднее профессиональное образование). - ISBN 978-5-16-016906-4. - Текст: электронный. - URL: https://znanium.com/catalog/product/1927269.