Контрольні запитання

1. Дайте визначення колекції. Наведіть типи колекцій

2. Наведіть основні інтерфейси, які успадковуються колекціями, та їх

призначення.

3. Поясніть призначення паттерну «Ітератор» та його реалізацію в .Net.

4. Дані якого формату зберігаються у хеш-таблицях, словниках? Які

переваги їх використання?

5. Поясніть призначення узагальнених типів. Наведіть приклади.

6. Поясніть призначення оператору default.

7. Поясніть призначення обмежень where в узагальнених типах. Наведіть

приклади.

8. Поясніть сутність нумератору колекції.

9. Наведіть способи опису нумератору в колекції, приклад створення

власного нумератору.

10. Роз’ясніть сутність поняття делегата.

11. Наведіть приклад опису делегата та виклику методу, використовуючи

делегат.

12. Наведіть склад класу делегату та поясніть, чим забезпечується

контроль типів в делегатах.

13. Поясніть сутність поняття анонімного методу.

14. Поясніть сутність лямбда-виразу, наведіть приклади лямбда-виразів.

15. Наведіть приклад опису події та генерування події.

16. Поясніть, яким чином виконується підписання на події та скасування

підписки.  
  
Відповіді:

1)Визначення колекції та типи колекцій:

Колекція - це об'єднання об'єктів одного типу, що забезпечує ефективний доступ до них і управління ними. Типи колекцій: Масиви, списки, словники, черги, стеки.

2)Основні інтерфейси колекцій та їх призначення:

IEnumerable: Надає інтерфейс для перебору елементів колекції.

ICollection: Визначає властивості і методи для додавання, видалення та отримання елементів колекції.

IList: Визначає інтерфейс для роботи з колекцією як зі списком.

IDictionary: Визначає інтерфейс для роботи з колекцією як із словником.

3)Призначення та реалізація ітератору в .NET:

Ітератор це паттерн, який надає механізм послідовного доступу до елементів об'єкта без викривання його внутрішньої реалізації. В .NET ітератори зазвичай реалізуються за допомогою ключового слова yield, яке дозволяє зупинятися і відновлювати виконання методу.

4)Формат даних: У хеш-таблицях і словниках дані зазвичай зберігаються у форматі "ключ-значення". Переваги: Швидкий доступ до даних за допомогою ключа, ефективність для пошуку, вставки та видалення елементів.

5)Призначення: Узагальнені типи дозволяють створювати класи, інтерфейси та методи, які працюють з різними типами даних, зберігаючи при цьому безпеку типів.

Приклади: List<T>, Dictionary<TKey, TValue>, Nullable<T>.

6)Оператор default визначає значення за замовчуванням для типу даних. Наприклад, default(int) поверне 0, а default(bool) - false.

7)where дозволяє вказати обмеження на тип, якій може бути використаний узагальнений параметр. Наприклад, where T : IComparable обмежує T інтерфейсом IComparable.

8)Нумератор - це об'єкт, який реалізує інтерфейс IEnumerator і використовується для послідовного перебору елементів колекції.

9)Реалізація інтерфейсу IEnumerator та IEnumerable. Створення власного класу, який імплементує ці інтерфейси та має методи для переходу до наступного елемента.

10)Делегат - це тип, який представляє собою посилання на метод. Він дозволяє передавати метод як параметр, зберігати його в змінній і викликати його пізніше

11) delegate int Example(int x, int y);

Example addDelegate = (a, b) => a + b;

int result = addDelegate(3, 5); // result = 8

12) Склад делегата включає в себе параметри та тип повернення методу, на який він вказує. Контроль типів забезпечується компілятором, який перевіряє, чи відповідає делегат сигнатурі методу.

13) Анонімний метод - це метод без імені, який може бути оголошений та використаний без визначення окремого методу.

14) Лямбда-вираз - короткий спосіб визначення анонімних методів. Приклад: (x, y) => x + y.

15) public class EventExample

{

public event EventHandler MyEvent;

public void RaiseEvent()

{

MyEvent?.Invoke(this, EventArgs.Empty);

}

}

16) EventExample example = new EventExample();

example.MyEvent += (sender, args) => Console.WriteLine("Event raised!");

example.RaiseEvent();

example.MyEvent -= (sender, args) => Console.WriteLine("Event raised!");