Изпитна тема № 5: Обектно-ориентирано програмиране

Итератори. Компаратори. Ламбда изрази и функции. Библиотека за обработка на колекции. Изключения. Работа с потоци и файлове.

Итераторите представляват методи, чрез които могат да се обхождат по специфичен начин различни структури от данни като масиви, списъци, опашки и т.н. Итератори използват израза “yield return”, за да върнат елементите един по един. При достигане на израза, при изпълнението на метода, се запазва мястото, до което е обходена съответната колекция от данни и изпълнението започва именно от това място при следващото извикване на итератора. Използването на итератори позволява използването на цикъл foreach при работа с колекцията. В следния пример е деклариран метод, който връща списък с всички нечетни числа, от 1 до подадено число, като параметър. Предимството на използването на итератор, е че при голямо подадено число, няма да има нужда да се изчаква зареждането на всички числа, а те ще се получават едно по едно, с всяко използване на итератора.

A screenshot of a computer code

Description automatically generated with medium confidence

Компараторите в C# се използват, когато трябва да бъде сортирана колекция от обекти или трябва да бъде извършено действие, което се нуждае от сравняване. Компараторите се реализират, чрез имплементиране на интерфейсът “IComparer<T>”. Това задължава в класа да бъде имплементиран метода Compare, който приема два обекта от тип “T” и връща цяло число, указващо резултата от сравнението. След като се реализира методът, според нуждите на програмата, той може да се използва като се предаде като аргумент на методите “Sort” или “OrderBy”. В следния пример е деклариран клас Singer, който има 2 свойства – Name и AlbumCount. След това е дефиниран компаратор, който сравнява обекти от тип Singer, спрямо полето AlbumCount.

A picture containing text, screenshot, font, line

Description automatically generatedA screenshot of a computer program

Description automatically generated with medium confidence

Ламбда изразите се използват за създаването на така наречените анонимни функции. Операторът за деклариране на ламбди – „=>“ се използва за разделяне на параметрите от тялото на ламбда израза. Всеки ламбда израз може да бъде превърнат в делегатен тип (delegate type). Типа на делегата зависи от това дали израза връща стойност и типа на параметрите. Ако ламбда изразът не връща стойност се използват делегати от тип Action<T1, T2…>, а ако връща стойност се използват делегати от тип Func<Т1, Т2…, TResult>. Има два основни вида ламбда изрази.

Изразни ламбда изрази (Expression lambdas)

A close up of a logo

Description automatically generated with low confidence

Операторни ламбда изрази (Statement lambdas)

A picture containing text, font, screenshot, line

Description automatically generated

Референциите към методи в C# са типове данни, които представляват указатели към методи. Те позволяват на програмата да работи с методи като обекти и да се предават като параметри или да се съхраняват в променливи. Референциите към методи се изразяват чрез делегати. Делегатите са типове данни, които декларират сигнатурата на метода, към който могат да се отнасят. Инстанция на делегат може да се създаде, като се укаже името на метода или се използва ламбда израз.

Колекциите са класове в C#, които предоставят по-гъвкав начин за работа с групи от обекти, в сравнение с масивите. За разлика от тях, при колекциите броят на елементите е динамичен и може да се увеличава според нуждите на приложението. Примери за колекции са List, Dictionary, LinkedList и др. Ако в колекцията има елементи само от един тип, трябва да се използва шаблонната версия на съответните класове, което ще предотврати добавянето на елементи от други типове. В следния пример е демонстрирана декларация на свързан списък от цели числа и отпечатването му на стандартния изход.

A picture containing text, screenshot, font, number

Description automatically generated

Language Integrated Query (LINQ) е подход/инструмент, който предоставя удобен начин за извличане и манипулиране на данни, без значение на техния източник, който може да бъде както база данни, така и колекция, съхранена в паметта. За работа с LINQ се използват ключовите думи ‘from’, ‘where’, ‘select’, ‘group by’, ‘order by’ и др. Тези ключови думи дефинират операциите, които трябва да бъдат извършени върху данните. Резултатът от изпълнение на LINQ заявка е нова колекция, която съдържа данните след прилагането и. В следния пример се демонстрира филтрация на данни от списък, чрез LINQ.

A screenshot of a computer program

Description automatically generated with medium confidence

Изключенията представляват аномалии или нередности, които възникват по време на изпълнение на програмата. Изключенията са обекти, които програмата връща/изхвърля (throw), в момента когато възникне грешка. В C# има вградени типове изключения като ‚Exception‘, ‚ArgumentNullException‘, ‚DivideByZeroException‘ и др. Въпреки това програмистите могат да създават и собствени типове изключения. Когато кодът хвърли изключение, изпълнението на програмата се прехвърля на блока за обработка на съответното изключение, а ако не бъде намерен такъв, изпълнението на програмата се прекратява. За работа с изключения се използва блоковете try, catch и finally. В блока try се поставя кода, който е възможно да хвърли изключение. В блока catch като параметър се поставя тип изключение, който се очаква и след това кода, който трябва да се изпълни при хвърляне на изключение. Блокът finally не е задължителен, а се използва при необходимост. Той се изпълнява независимо от това, дали в блока try е възникнало изключение.

Файловете и потоците са основните концепции при работата с входно-изходни операции в програмирането. Файловете представляват именувано място от диска, в което може да бъде съхранявана информация от различни видове. Потоците представляват последователност от данни, от която може да се чете информация или да се записва такава. В програмирането потоците се използват за пренос на данни между програмата и външни източници като файлове, конзолата и др. Чрез потоците може да се чете част по част, което е ефективно при работа с големи обеми от данни. За да работи една C# програма с файлове първо трябва да бъде отворен поток, който да бъде свързан с него. За целта се използва класовете: „FileStream“, “StreamReader” и “StreamWriter”. При създаване на инстанция се посочва адреса на файла, към който трябва да бъде отворен потока. За четене от файл се използват методите „Read“ и „ReadLine“. За записване на информация във файл се използват методите “Write” и “WriteLine”. След приключване на работа с файла, потока трябва да бъде затворен, за да се освободят ресурсите. Това се случва с използването на метода “Close” или чрез използването на блокове „using“, които автоматично ще затворят потока, след изпълнение на блока.

Обобщен план (Опорни думи)

1. **Итератори**
   * Методи за обхождане на структури от данни
   * ‘yield return’ за връщане на елементите един по един
   * Запазва мястото, до където е обходена колекцията
   * Позволява foreach
   * Пример – 1…N, връща само нечетни, предимства голямо число, не изчаква обработката на всички
2. **Компаратори**
   * Използват се, когато трябва да се сортират или сравнят обекти
   * Имплементиране на IComparer<T>
   * След това инстанция на класа се ползва в методите Sort и OrderBy
   * Пример – клас Singer, клас AlbumComparer, сравнява броя на албумите, сортиране по възходящ ред.
3. **Ламбда функции и изрази**
   * Създаване на анонимни функции
   * => - оператор за деклариране на ламбди, разделя параметри от тяло
   * Всяка ламбда може да се превърне в делегатен тип, ако не връща стойност Action<T1,T2…>, ако връща стойност, Func<T1, T2…, TResult>
   * Expression Lambdas – Func<int,int> square = x => x\*x
   * Statements Lambdas – цял израз, Пример – приема параметър string и печата Hello name на конзолата
   * Указатели към методи – позволяват работата с методи като обекти, предава се като параметри или се запазват в променливи. Изразяват се, чрез делегати. Делегатите дефинират сигнатура на метода. Посочва се или име на метода или ламбда израз.
4. **Библиотека за обработка на колекции**
   * Колекции – по-гъвкав начин за работа с група от обекти, динамичен размер, List, Dictionary, Queue и т.н. Ако е един тип, generic версията
   * Language Integrated Query (LINQ) – удобен начин за извличане и манипулиране на данни, без значение на техния източник. Ключови думи ‘from’, ‘where’, ‘select’, ‘group by’, ‘order by’ и др. Резултата е нова колекция. Пример – списък с имена, взимане само тези, които имат ‚а‘
5. **Изключения**
   * Аномалии или нередности, възникнали по време на изпълнение
   * Изключенията се обекти, вградени типове ‚Exception‘, ‚ArgumentNullException‘, ‚DivideByZeroException‘ и др. Има възможност за custom типове.
   * При изключение изпълнението се прехвърля в блока за обработка
   * try – кода, който може да хвърли грешка
   * catch – параметър, типа на грешката, кода който се изпълнява при изключение
   * finally – по избор, изпълнява се независимо дали има изключение
6. **Работа с потоци и файлове.**
   * Файлове- именувано място на диска, съхраняване на инф. от различни видове
   * Потоци- последователност от данни, от която може да се чете и пише. Пренос на данни между програмата и външен източник – файл, конзола. Четене част по част
   * За работа с файл в C# се отваря поток, чрез класовете „FileStream“, “StreamReader” и “StreamWriter”. При инстанциране се посочва адреса на файла. След приключване на работа се затваря потока. Използване на using блок, затваря потока сам след изпълнение.

Казус

Извикайте функциите Add(), Subtract(), Multiply() и Divide(), като използвате multicast делегати.

A screenshot of a computer code

Description automatically generated with low confidence