Елементи от функционалното програмиране

1.Ламбда издрази:

Ламбда изразът е анонимна функция, която осигурява кратък и функционален синтаксис, който се използва за писане на анонимни методи. Той се основава на концепцията за програмиране на функции и се използва за създаване на делегати или типове дървета на изрази. Синтаксисът е функционален (arg1, arg2 … argn) израз. Ламбда изразите използват оператор символ =, който се чете като "отива". Входните параметри са посочени от лявата страна на оператора, а декларацията / изразите са посочени от дясната страна. Обикновено ламбда изразите не се използват директно в синтаксиса на заявките, но често се използват в обаждания на методи. Изразът на заявките може да съдържа обаждания на методи.

Характеристиките на синтаксиса на Lambda за изразяване са следните:

Това е функция без име.

Няма модификатори, като претоварвания и превишения.

Тялото на функцията трябва да съдържа израз, а не израз.

Може да съдържа повикване към процедура на функция, но не може да съдържа повикване към подпроцедура.

Декларацията за връщане не съществува.

Стойността, върната от функцията, е само стойността на израза, съдържащ се в тялото на функцията.

Операторът за функция End не съществува.

Параметрите трябва да имат определени типове данни или да се правят изводи.

Не позволява общи параметри.

Не позволява опционални и ParamArray параметри.

2.Ламбда функции:

Просто казано, ламбда функцията е точно като всяка нормална функция на Python, с изключение на това, че няма име, когато се дефинира, и се съдържа в един ред код. Ламбда функцията оценява израз за даден аргумент. Давате на функцията стойност (аргумент) и след това предоставяте операцията (израз).

3.LINQ:

Езикова интегрирана заявка (LINQ) е името на набор от технологии, базирани на интегрирането на възможности за заявки директно в езика C#. Традиционно заявките към данни се изразяват като прости низове без проверка на типа по време на компилиране или поддръжка на IntelliSense. Освен това трябва да научите различен език за заявки за всеки тип източник на данни: SQL бази данни, XML документи, различни уеб услуги и т.н. С LINQ заявката е първокласна езикова конструкция, точно като класове, методи, събития. Вие пишете заявки срещу строго типизирани колекции от обекти, като използвате езикови ключови думи и познати оператори. Семейството от технологии LINQ осигурява последователно изживяване при заявки за обекти (LINQ към обекти), релационни бази данни (LINQ към SQL) и XML (LINQ към XML).

За програмист, който пише заявки, най-видимата „интегрирана в езика“ част от LINQ е изразът на заявката. Изразите на заявката са написани в декларативен *синтаксис на заявка* . Като използвате синтаксис на заявка, можете да извършвате операции за филтриране, подреждане и групиране на източници на данни с минимален код. Използвате едни и същи основни шаблони на израз на заявка, за да заявявате и трансформирате данни в SQL бази данни, ADO.NET набори от данни, XML документи и потоци и .NET колекции.

Можете да пишете LINQ заявки в C# за бази данни на SQL Server, XML документи, ADO.NET набори от данни и всяка колекция от обекти, която поддържа IEnumerable или общия интерфейс IEnumerable<T> . Поддръжката на LINQ се предоставя и от трети страни за много уеб услуги и други реализации на бази данни.

4.Делегати и функционално програмиране:

В компютърните науки функционално програмиране е парадигма за програмиране – стил за изграждането на структурата и елементите на компютърни програми, който третира като изчислява оценката на математически функции и избягва променящите състоянието непостоянни данни. Това е декларативна парадигма за програмиране, което означава, че програмирането се извършва с изрази. Във функционален код, изходната стойност на функция зависи само от аргументите, които са вложени по време на функцията, така призованата функция е два пъти с една и съща стойност за един аргумент Х и тя ще произведе същия резултат (Х) всеки път. Премахване на нежелани реакции, т.е. промени в състоянието, което не зависи от входовете на функцията, може да направи много по-лесно разбирането и да се предскаже поведението на една програма, което е един от основните мотиви за развитието на функционално програмиране.

Функционално програмиране има своите корени в ламбда пресмятането, което е официална система, разработена през 1930 г., за да разследва изчислимост на дефиниция на функция, приложение функция и рекурсия. Много функционални езици за програмиране могат да бъдат разглеждани като изградени върху ламбда пресмятането. Друга добре позната декларативна парадигма за програмиране е логическото програмиране което, се основава на релацията.

За разлика, императивното програмиране променя състоянието, чрез команди в началния език, най-прост пример за това е задачата. Императивното програмиране наистина има функции, не в математическия смисъл, а в смисъл на подпрограми. Те могат да имат странични ефекти, които могат да променят стойността на програмата. Функции без стойности за връщане могат да доведат до това. Поради това, че те нямат референтна прозрачност, т.е. същия израз в друг език може да доведе до различни стойности по различно време в зависимост от състоянието на изпълняващата програмата.

Функционални програмни езици, особено изцяло функционални такива като Hope и Rex, са използвани повече в академичните среди, отколкото в развитието на търговски софтуер. Въпреки това, видни функционални езици за програмиране като Common Lisp, Scheme, Clojure, Language Wolfram Language, Racket, Erlang, OCaml, Haskell, и F# са били използвани в промишлени и търговски заявления от голямо разнообразие от организации. Функционално програмиране се подкрепя и в някои специфични програмни езици за домейн като R, J, K и (programming language from Kx Systems) Q от Kx Systems (финансов анализ), XQuery / XSLT (XML), и Opal. Широко разпространени специфични програмни езици за домейн като SQL и Lex / Yacc използват някои елементи на функционалното програмиране.

Програмиране на функционален стил може да се осъществи в езици, които не са специално предназначени за функционално програмиране. Например, императивния Perl език за програмиране е бил обект на една книга, описваща как да се прилагат концепции на функционалното програмиране. Това е вярно и за езика PHP. C # 3.0 и Java 8 добавят конструкции за улесняване на функционалния стил. Езикът Julia също предлага функционални възможности за програмиране. Интересен случай е този на Scala – често се пише във функционален стил, но присъствието на странични ефекти и непостоянни състояния го провежда в сивата зона между императивни и функционални езици.