ZÁRÓDOLGOZAT

Tóthné Ondrék Marianna

2024

Magyar Agrár- és Élettudományi Egyetem

Károly Róbert Campus

Programtervező informatikus felsőoktatási szakképzés szak

C# LINQ ÉS LAMBDA KIFEJEZÉSEK AZ EXCEL ADATOK LEKÉRDEZÉSÉBEN ÉS MŰVELETEIBEN

Belső konzulens:	Dr. Pántya Róbert
	Egyetemi adjunktus
Belső konzulens intézete/tanszéke:	**Magyar Agrár- és Élettudomány Egyetem/ Műszaki intézet**
Külső konzulens:	-
	-
Készítette:	**Tóthné Ondrék Marianna**

Gyöngyös

2024

Tartalomjegyzék

1. Bevezetés

A záródolgozatomat a lambda és LINQ kifejezésekből írom a C# és Excel programokon keresztül.

Ez a dolgozat praktikus példákat mutat be arra, hogyan használhatja a nem programozó a C# LINQ-ot és lambda kifejezéseket az Excel-adatok lekérdezéséhez, szűréséhez, rendezéséhez és elemzéséhez.

A C# LINQ és lambda kifejezések segítenek az Excel-ben történő automatizálásra és hatékonyabbá tételére, amivel a felhasználók időt takarítanak meg. Ezt az időt fontosabb dolgokra használhatják.

Felhasználása széleskörű: pl.: listák szűrésére, rendezésére, vagy a keresett elemek kinyerésére, számokat szöveggé alakításra, események kezelésére pl.: button clic.; betűrendbe rendezésre.

Az IT-ban ezeknek az alkalmazása a felhasználás csúcsát jelenti. Továbbá elmondhatjuk, hogy mind a C#, mind az Excel használatában a felhasználók ismerik a saját szintjükön. Viszont nem mindenki alkalmazza naprakészen. Tehát mindannyian ismerjük az Excelt, van aki egyszerű táblázatokat formáz, mint egy órarend, és van aki összetettebb folyamatokra használja, függvények képletek, akár makró segítségével összetett lekérdezések, változók megadásának használatára pénzügyi és ipari területeken.

A világ fejlődésével az elvárás folyamatos. Hangsúlyt kell fektetni a munka logikai felépítésére, gyorsaságára és alkalmazhatóságára. Korábbi munkám során amikor látták, hogy Excellel dolgoztam, a többiek azt mondták, hogy mit képzelek én, az már egy elavult darab. Úgy gondolom, hogy ha a Google, az Amazon vagy a Microsoft számára megfelelőek, akkor még nyújthatnak haladó, fejlett megoldásokat további vállalatoknál is, és akár nekem is.

A C# nyelv és az Excel továbbra is népszerűek a vállalkozások és szervezetek körében. A C# rugalmas programozási nyelv, amely alkalmas webes és asztali alkalmazások fejlesztésére. Az Excel pedig egy hatékony táblázatkezelő program, amely széles körű funkciókkal rendelkezik az adatok elemzéséhez és vizualizálásához. A C# LINQ és lambda kifejezések használata ezekkel a szoftverekkel további hatékonyságot és termelékenységet biztosíthat. A Visual Studio-nak pedig olyan nagy a nyelvtámogatása, ami egyedülállóvá teszi a programozási környezetek között. Ilyenek pl: C#, Visual Basic.NET, C++, F#, Java, Python, JavaScript.

Azért szimpatikus számomra még ezek a Microsoft alkalmazások, mert részletes leírásokkal, videókkal támogatják az alkalmazás megtanulását. Tehát megtanulhatók.

A lambda, LINQ segítségével a munkafolyamataim rövidebbek. A lekérdezéseim egyszerűbbek átláthatóbbak.

Azzal, hogy ezek a nagyszerű programok idáig jutottak a fejlesztésben, segítséget nyújtanak a mindennapjainkban, azáltal, hogy a szükségleteink kielégítésére nem kell egy másik programot fejlesztenünk ahhoz. Hanem egy jól felépített program nyújt támogatást folyamatos fejlesztéssel és karbantartással.

2. Szakmai gyakorlat

A szakmai gyakorlatomat a **Rosenberger Automotive Cabling Kft**.-nél töltöttem. A Rosenberger Automotive Cabling szakértője az autóipari kommunikációs rendszerek kábelszerelvényeinek gyártásának. Termékeiket a járművek legmodernebb információs rendszereiben használják, és összekapcsolják a különböző elektronikus alkatrészeket.

Rádiófrekvenciás és érzékelő kábeleket szerelnek össze a megrendelő igényei szerint. De nem csak a gyártásban, hanem már a fejlesztési fázisban is a Rosenberger Automotive Cabling minősített partner.

1. ábra Fotó (forrás: *-https://www.rosenberger.com/company/rosenberger-worldwide/rosenberger-automotive-cabling/*)

„100%-ban ügyfeleink igényeire összpontosítunk, erősségünk a minőség és a megbízhatóság.”

(Rosenberger Automotive Cabling honlapja. Letöltés dátuma: 2024.04.16, forrás: https://www.rosenberger.com/company/rosenberger-worldwide/rosenberger-automotive-cabling)

2.1 A Rosenberger vállalat bemutatása

A Rosenberger az autóipari csatlakozók és száloptikás rendszerek specialistája. Rendelkezésre áll a tömeggyártás, és az egyedi igények kielégítésre. A nagy mennyiségű műanyagipari alkatrészei, fémipari kábelei és csatlakozói világ szinten elterjedtek Tajwantól egészen Dél-Amerikáig. Ez a 60 éves múlttal rendelkező vállalat Magyarországon 6 telephellyel rendelkezik, összesen 3000 munkavállalója van és elmondhatom, hogy a Rosenberger név világ szinten jegyezve van az autóipari piacon.

(A Rosenberger hivatalos honlapja. Letöltés dátuma: 2024.04.16, forrás: *https://www.rosenberger.com/company/history/*)

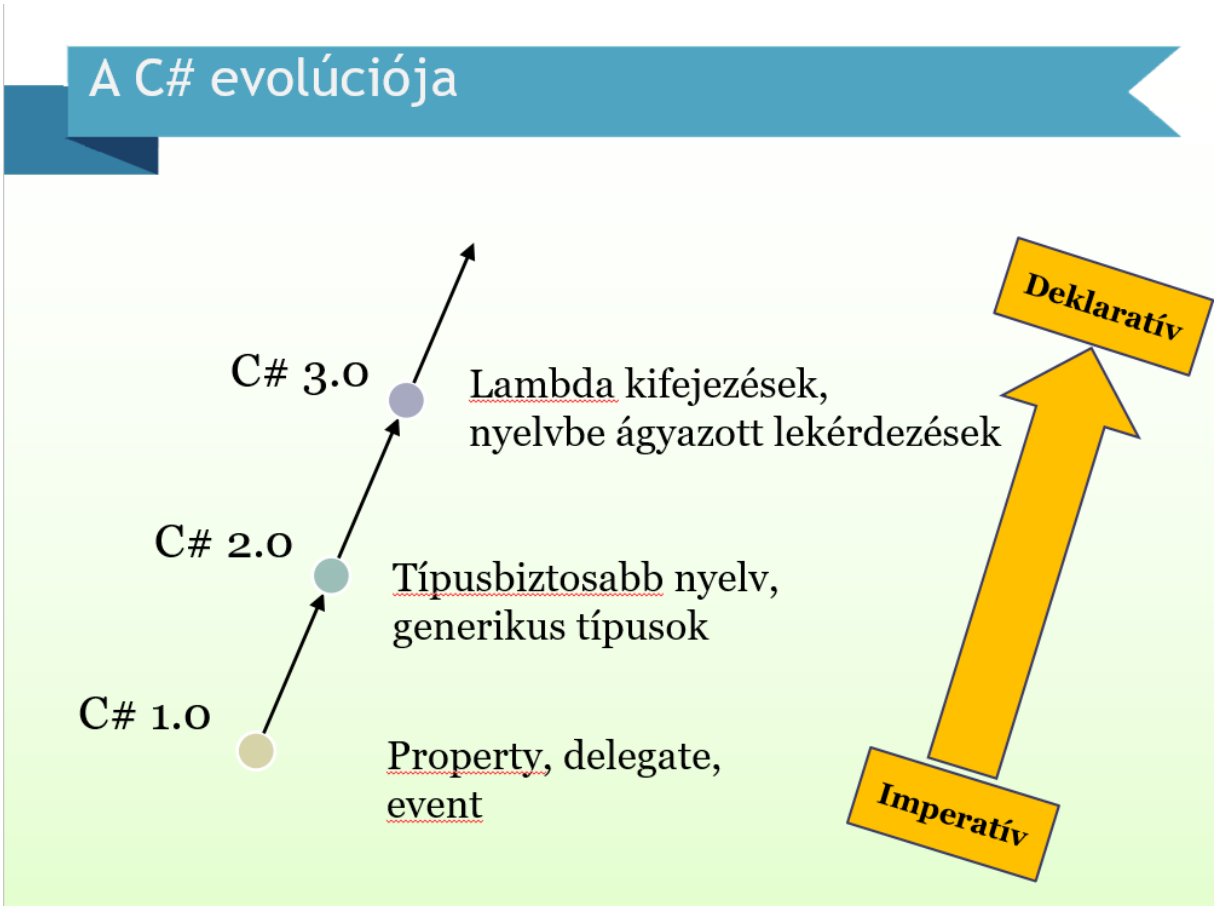
Munkám során a minőségügyi csoport munkáját segítettem, a formanyomtatványok digitális működtetésével.

A szűk felhasználási lehetőségek miatt Excel makró alkalmazása a legegyszerűbb és legkézenfekvőbb.

3. Szakirodalmi áttekintés

2. ábra: A C# evolúció

(forrás: *Lambda kifejezések és Linq: Magasszintű programozási nyelvek II. Kovásznai Gergely, Pányta Róbert.)*



3.1 A C# nyelv

3.1.1 2002- C# 1.0:

-A 2002-ben kiadott verziója a Java-hoz volt hasonló. A keretrendszerből még hiányoztak a ma ismert tulajdonságok. Hiányoztak belőle a ma ismert aszinkron képességek és az általunk ma már jól ismert általános funkciók. Ekkor még hiányoztak a generikusok és a LINQ.

Főbb jellemzői voltak:

- osztályok
- Struktúrák
- Interfészek
- Események
- Tulajdonságok
- Delegate-ek
- Operátorok és kifejezések

- Attribútumok

3.1.2: 2005- C# 2.0:

Megjelennek a generikusok. Az általános OO (Objektum Orientált) programozási nyelv típusossá válik. A generikusok segítségével a típusok és módszerek tetszőleges típuson működhetnek úgy, hogy meg őrizik a típusbiztonságot. A generikusok általános típusok.
Pl.: List<T> lehetővé teszi, hogy típusbiztos műveleteket hajtson végre.
A C# 2.0-s verziójával az iterátorok megjelentek a nyelvben. Az iterátorok olyan objektumok a nyelvben amiknek a segítségével tudunk végig menni a kódon. Egyetlen FOREACH parancs lehetővé teszi a ciklusban, hogy a vizsgálat minden elemen végre legyen hajtva.
A 2.0 elhozta a nyelvnek azt az áttörést, hogy a kód olvashatósága egyszerűbb legyen a fejlesztők számára.
Főbb jellemzők:

- Generikusok
- Részleges típusok
- Névtelen módszerek
- Nullálható értéktípusok
- Iterátorok
- Kovariancia/ kontravariancia
- Getter-Setter
- Módszercsoport konverziók (küldöttek)
- Statikus osztályok
- Delegált következtetés

3.1.3 2007-C# 3.0

A C# 3.0 érte el az áttörést a lekérdezési funkciók hozzáadásával. Itt került bevezetésre a LINQ más néven Language-Integrated Query. Ebben a verzióban a kifejezésfákat, lambda-kifejezéseket és az anonim típusokat vizsgálja ez a hibrid objetum-orientált / funkcionális nyelv.
Ez a verzió lehetővé teszi, hogy a kód SQL- stílusú lekérdezéseket hajtson végre.
Pl. A számok átlagának kiszámításához egy FOR ciklus megírása helyett elegendő egy list.Average() metódus.
Főbb jellemzők:

- Automatikus megvalósított tulajdonságok
- Névtelen típusok
- Lekérdezési kifejezések

- Lambda kifejezések (λ)
- Kifejezési fák
- Bővítési módszerek
- Implicit módon beírt változók
- Részleges módszerek
- Objektum- és gyűjtemény inicializálók

(Dietrich E.-Samacchia P. (2017): NDepend blog alapján, *C# version history* Letöltés dátuma: 2024.04.16. forrás: *https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history#c-version-10-1*)

3.2 Hogy mennyire fejlődőképes nyelvről beszélünk?

A C# 3.0 óta összesen 12 verzióval bővült a keretrendszer. Ebből csak a tanulmányaim során jelent meg 2.

A lambda kifejezések fejlesztése a 9. 10 és 12. verzióban megjelent.

3.2.1 2020- C# 9.0

Megjelenik a .NET5. A számos funkció mellett, a lambda-eldobási paraméterekkel fejlesztve lett.

Lehetővé teszi, hogy az eldobott (_) karaktereket lambdák és anonim metódusok paramétereiként használják. Például:

lambdas: (_, _) => 0,(int _, int _) => 0

névtelen módszerek:delegate(int _, int _) { return 0; }

Vagyis – A lambda, vagy névtelen metódusok paraméter listájában több paraméterrel ezek eldobott paraméterek.

3.2.2 2021- C# 10.0

Fejlesztések a C# nyelvben.

A lambda kifejezések javítása: Itt a kifejezések már természetes típusúak lehetnek. Deklarálhatnak visszatérési típust. Ez segít abban, ha a fordító nem tud következtetni.

Nincs szükség delegált típusú változó deklarálására.

3.2.3 2023- C# 12.0

Funkciók hozzáadása a C#-hoz.

Többek között az Opcionális-Lambda kifejezések a C# nyelvben. Innentől meg lehet adni alapértelmezett kifejezéseket a lambda +- kifejezések paramétereihez.

A lambda kifejezések főbb szabályai:

- A lambda kifejezés mindig névtelen függvényt használ.
- A lambda kifejezés két formája ismert:

- Amikor a teste egy kifejezés
- Amikor a törzse egy utasítás blokk

- Bal oldalon találhatók a bemeneti paraméterek, a másik oldalon pedig az utasítás blokkok.
- Bármelyik lambda kifejezés lehet delegált típusú
- Lambda kifejezések konvertálása történhet ACTION vagy FUNC delegált típusra. Actionra való konvertálás akkor történik, ha nem ad vissza értéket.
- Lambda kifejezések használhatók ahol delegált típusok és kifejezésfák szükségesek.
- Továbbá használhatók a LINQ kifejezéseknél.

Az operátor jobb oldalán levő kifejezéseket, lambda kifejezéseknek nevezzük. „=>”

- A lambda utasítás: hasonlít a lambda kifejezéshez. Tartalma kapcsos zárójelek közé van téve. pl:” (input-parameters) => { <sequence-of-statements> }”
- Törzse: Tetszőleges számú utasításokból áll.
- Kifejezésfák létrehozására nem használható lambda utasítás.
- A lambda kifejezés bemeneti paraméterei:
  - 0 vagy üres érték: ()
  - Egy bemeneti paraméter esetén: zárójel használata nem kötelező
  - Két, vagy több bemeneti paramétert, vesszővel kell elválasztani: (x, y, z)
  - A fordító nem mindig tud egyértelműen következtetni a paraméterek megfelelő típusára. Ezért azok a kódban bátran megadhatók. pl: Func < int, double, string>
  - A bemeneti paramétertípusoknak mindig explicitnek, vagy implicitnek kell lenniük, ellenkező esetben hiba lép fel.
  - Az eldobások „_, _” segítségével kettő, vagy több paraméter is megadható.
  - A C# 12.0-tól kezdve alapértelmezett értékek is megadhatók.
  - Egy tömbparaméterrel rendelkező metóduscsoport „params” is hozzárendelhető egy lambda kifejezéshez.
- A lambda kifejezés argumentumaként megadható egy sor, és maga a lambda kifejezés is visszaadhat egy sort.
- A sor meghatározható a zárójelben szereplő, vesszővel tagolt listájával.

(Gibson R. 2016-2024 GitHub alapján, *The history of C#* Letöltés dátuma: 2024.04.16, forrás: *https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history#c-version-10-1*)

3.3 LINQ = Language Integrated Query.

„A lekérdezési képessége, közvetlenül a C# nyelvbe történő integrációján alapuló technológiai halmaz neve.”

Az adatokkal kapcsolatos lekérdezéseket, egyszerű karakterláncokkal fejezi ki.

Minden adatforrástípushoz más lekérdező nyelvet kell megtanulni:

- SQL adatbázisok
- XML dokumentumok
- egyéb web szolgáltatások

A LINQ nyelvbe integrált része a lekérdezési kifejezés. A lekérdezési kifejezések deklaratív lekérdezési szintaxisban íródtak. Ezekkel a szintakszisokkal szűrési, rendezési és csoportosítási műveleteket lehet végrehajtani.

3.4 Lekérdezési kifejezések

A lekérdezési kifejezések alkalmasak mind lekérdezésekhez, mind átalakításokhoz, amelyeknek az adatforrása kompatibilis a LINQ-val.

Például lekérdezhetünk SQL adatokat a LINQ segítségével, majd XML formátumban állíthatjuk elő.

A lekérdezési változók sok ismerős nyelvi kifejezést használnak, hogy megkönnyítsék az olvashatóságot és erősen kacsolódnak a lekérdezési kifejezésekhez.

A lekérdezések futtatásához integrációra van szükség pl. FOREACH segítségével.

A C#-ban a fordításkor a lekérdezési kifejezések lekérdezési operátorokká válnak.

Ami lekérdezés szintaxissal kifejezhető, az bármilyen metódus szintaxissal is kifejezhető.

Lekérdezések azok a kifejezések, amelyek adatokat kérnek le az adatforrásból.

A különböző adatforrásokhoz különböző lekérdezési nyelvek tartoznak.

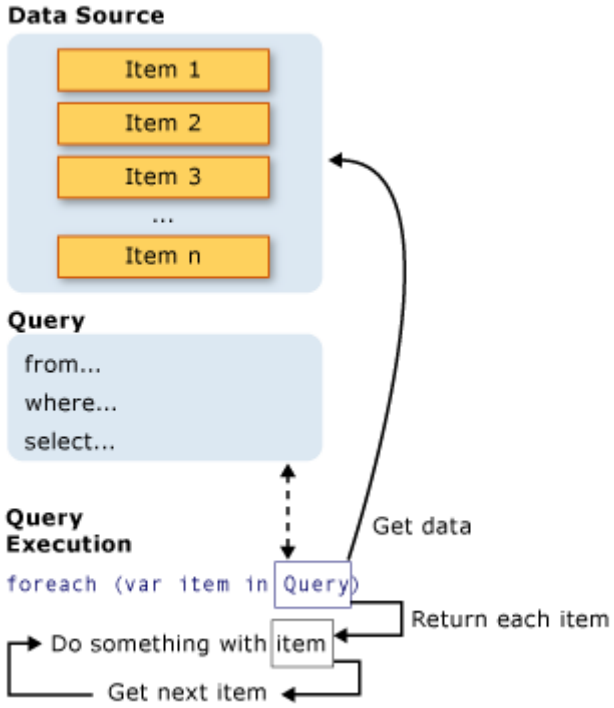
A LINQ konzisztens C# nyelvi modellt kínál a különböző adatforrásokhoz és formátumokhoz.

A LINQ lekérdezésekben mindig C# objektumokkal dolgozunk.

A lekérdezési művelet három részből áll:

1. Adatforrás megszerzése
2. Lekérdezés létrehozása
3. Lekérdezés végrehajtása

3. ábra: Lekérdezési műveletek (forrás*: https://learn.microsoft.com/en-us/dotnet/csharp/linq/)*



A LINQ adatforrása általában tömb. Az adatforrásnak nincs szüksége bármilyen testre szabásra a lekérdezéshez.

3.5 A lekérdezés

A lekérdezésekkel meghatározhatjuk, hogy milyen információkra van szükségünk az adatforrásból.

Ezekről eldönthetjük, hogy rendezve, csoportosítva, formázva kapjuk meg a lekérdezéseink során.

A lekérdezések során a C# szintaxisok vannak a segítségünkre.

Pl. A lekérdezés a tömb összes páratlan számát adja vissza sorrendben.

A lekérdezéshez három kulcsszó szükséges: FROM=adatforrás, WHERE=szűrő, SELECT=típus

Fontos, hogy ezek a lekérdezések nem végeznek semmilyen műveletet, csak végrehajtják a parancsot.

3.6 Azonnali végrehajtás kényszerítése

Azok a lekérdezések amelyek aggregációs funkciókat hajtanak végre, az összes elemen végig kell menniük. Ilyenek pl.: COUNT, MAX, AVERAGE, FIRST. Ezek az utasítások FOREACH parancs nélkül lefutnak.

A végrehajtást lehet kényszeríteni úgy is, hogy közvetlenül a FOREACH ciklus után fussanak le.

(The history of C#, 2023.12.15. *Overview of LINQ*, Letöltés dátuma: 2024.04.16, forrás: https://learn.microsoft.com/en-us/dotnet/csharp/linq/)

A lekérdezés olyan utasításkészlet, amely leírja, hogy milyen adatforrásból, milyen adatokat kell lekérni meghatározott alakban.

Általában a forrásadatok logikailag sorrendben vannak rendezve.

Pl. Az SQL táblában, vagy az XML-fájl-ban az adatok elemek, sorba vannak rendezve.

A lekérdezések háromféle módon mehetnek végbe:

- Az elemek részhalmazának lekérdezése
- Az elemek sorozatának lekérdezése
- Egyedi érték lekérdezése

3.6.1 Lekérdezési kifejezés

A lekérdezési kifejezés első osztályú nyelvi konstrukció. Bármilyen C# környezetben használható.

Jellemzőjük, hogy FROM kifejezéssel kezdődnek, és SELECT, vagy GROUP paranccsal záródnak.

3.6.2 Lekérdezési változó

A LINQ-ban minden olyan változó, ami lekérdezést tárol.

Egy felsorolható típus, ami elemek sorozatát állítja elő.

3.6.3 Var kulcsszó

A var kulcsszó használata lehetővé teszi a kódnak, hogy következtessen a lekérdezési változó típusára.

3.6.4 Lekérdezési kifejezések

- FROM: a lekérdezési kifejezés kezdete
- SELECT: a lekérdezési kifejezés befejezése
- GROUP: csoportos lekérdezés
- ORDERBY: elemek újra rendezése
- INTO: további lekérdezések kezdeményezése
- WHERE: szűrés

- JOIN: két vagy több adatforrásból származó elemek társítása
- LET: legyen, egy eredmény új tartományváltozóban való tárolására

(Wagner B.-Klonowski B. *Query expression basics*, 202.03.06. Letöltés dátuma: 2024.04.16, forrás: https://learn.microsoft.com/en-us/dotnet/csharp/linq/)

3.7 Az Excel

Az Excel 1985-ben lett kifejlesztve az Apple számára. A Microsoft számára csak 1987-ben került bevezetésre.

A létrehozása négy fiatalnak köszönhető: *Doug Klunder, Jabe Blumenthal, Charles Simonyi és Bill Gates.*

A táblázatkezelés elméleti alapjait már korábban megfogalmazta *Richard Mattesich* (a californiai Berkeley Egyetem tanára) 1964-ben *„Simulation of the Firm Through a Budget Computer Program”* c. könyvében.

Az Excel sikerének a megalapozása egy olyan új szoftver fejlesztése, ami kompatibilis a Macintosh rendszerével, teljesen elvonatkoztatva a korábbi MS-DOS programoktól és jobb, gyorsabb mint a korábbi programok.

Az igazi áttörést az „intelligens újraszámlálás” funkció hozta el.

Ami kezdetben úgy működött, hogy amikor egy cella megváltozott, az Excel inkább kiválasztotta azokat a cellákat, amire a változás hatással volt. Minthogy újra számolta volna az egészet.

A funkción dolgozó csapatnak annyira megtetszett, hogy az alábbi feliratú pólókat kezdtek viselni: „Excel – Számold újra vagy halj meg (Excel – Recalc or Die)”

4. ábra: Számold újra, vagy halj meg

(forrás: *https://sosexcel.com/exceltippek/az-excel-tortenete/)*

Az Excel első kiadása 1985.09.30-án volt, ahogy a tervezték, bár nem volt teljesen zökkenőmentes. Az utolsó éjszakán még átellenőrizték a kódokat, és ahol szükséges volt, javításokat végeztek.

A Microsoft-ra készült verziót csak 1987-ben adták ki.

(Kőrösi B. *Az Excel születése – a kiválóság* története!* (2019), Letöltés dátuma: 2024.04.16. forrás: https://sosexcel.com/exceltippek/az-excel-tortenete/)

Az Excel 5.0 hozta el 1993-ban a programozási támogatást, a VBA alapú folyamatok automatizálásával.

A 12.0 verzióban jelenik meg az Open XML formátum.

(N. a. *Wikipedia: Microsoft Excel*, Letöltés dátuma: 2024.04.16, forrás: *https://hu.wikipedia.org/wiki/Microsoft_Excel*)

Az Excel lambda 2020 decemberében került bejelentésre.

A lambda új lehetőségeket ad a felhasználók számára, ezzel az Excel-turing teljessé vált.

Gyakorlatilag innentől bármilyen függvényt lehet írni.

(Gordon A.-Jones S. P.(2021)*LAMBDA: The ultimate Excel worksheet function*, Letöltés dátuma: 2024.04.16, forrás: https://www.microsoft.com/en-us/research/blog/lambda-the-ultimatae-excel-worksheet-function/)

Az Excel csapata bejelentette a lambda-t, egy olyan új funkciót amely lehetővé teszi a felhasználók számára, hogy egyéni képletfüggvényeket hozhassanak létre, amelyek hasonlóan viselkednek mint az Excel függvények. A lambda függvények, paraméter függvényeket fogadnak el, meghívhatnak más lambda függvényeket és rekurzívan hívhatják meg saját magukat.

(N. a. InfoQ, Az Excel képletnyelve immár Turing-teljes, Letöltés dátuma: 2024.04.21, forrás: https://www.infoq.com/articles/excel-lambda-turing-complete/)

4. Alkalmazott módszerek

A vizsgálat célja, hogy összehasonlítsam a két szoftvert, és bebizonyítsam, hogy nem szükséges professzionális tudás a lekérdezések elvégzésére. Hogyan lehet elvégezni ugyanazokat a lekérdezéseket lambda kifejezéssel a két szoftverben?

Az Excelben a lambda kifejezéssel a lekérdezés ugyanúgy elvégezhető, mint C#-ban.

A vizsgálat során a Visual Studio-ban C# nyelven megírtam a programot, és ugyanazokkal az adatokkal készítettem Excel lekérdezéseket. Az összehasonlítás eredménye az, hogy mindkét program azonos eredményt hozott, annak köszönhetően, hogy lehetséges volt ugyanazt a funkciót ellátni. A lekérdezésekben, az általam megadott adatokat használtam fel, pont úgy ahogyan a valóságban is feltöltünk például egy Excel táblázatot adatokkal a munkahelyünkön. Vagy megkapunk egy nagyméretű táblázatot adatokkal, annak érdekében, hogy kimutatásokat készítsünk a vezetőség számára egy beszámolóra. Ezek az adatok nem véletlenszerűen kerültek be a rendszerekbe, mivel a véletlen adatok megadása nem részei a feladatomnak. A vizsgálatomban a lekérdezések során a lambda kifejezésekkel együtt az átlag függvényt, a szűrő függvényt, a keres függvényt, és a ha függvényt használtam. A Visual Studio-ban a VAR kulcsszó segített abban, hogy a program automatikusan tudjon következtetni. Ezzel a kód egyszerűbb és olvashatóbb.

A Visual Studio-ban a C# kódok megírásához szükséges programozási alapismeret. Az Excelben, a lambda-nak hála, nem kell programozási alapismeretekkel rendelkeznünk ahhoz, hogy könnyen és gyorsan írjunk lekérdezési függvényeket. Ennek azért van jelentősége, mert nagy méretű Excel táblákat üzleti, banki és ipari felhasználásra használnak. Az ilyen jellegű területeken nem elvárás a programozói ismeret, ezek a metódusok az ő munkájukat segíti. Azon felül, hogy a lambda kifejezések hozzájárulnak nagy mennyiségű adatok feldolgozására, a kutatásban hozzájárulnak a gépi tanuláshoz, a neurális hálózatokhoz ami lehetővé teszi, hogy a modellek megtanulják az adatokban rejlő mintákat. Tehát segítséget nyújtanak az automatizálásban, az orvostudományban, a gyártásban, a kereskedelemben és a közlekedésben.

5. Eredmények és értékelésük

A szakmai gyakorlatom során, az Excel volt a fő eszközöm a vállalatnál felmerülő informatikai problémák megoldására.

Egy nagyvállalat számos olyan adattal rendelkezik ami a napi tevékenységét méri. Ezekhez a mérésekhez, mérőszámok kapcsolódnak. A mérőszámokat KPI-nak nevezzük. A KPI számok megadásához a gyárban működő összes részleg Excel táblázatban gyűjti az adatait. Pl:. gyártott mennyiség, munkaórák száma, selejt mennyisége, jelenlét, balesetmentes napok száma, kiszállított mennyiség stb.

Ehhez elengedhetetlen az Excel használata. Azon belül is a program haladó ismerete. Szerencsére ma már az Excel számos megoldást nyújt az egyszerű felhasználástól a professzionális szintig.

Mit nyújt nekünk az Excel? Cellákat a nagy mennyiségű adatokhoz, és függvény készletet az adataink feldolgozásához.

Hogy hányféle függvény létezik? Több száz.

Bár az Excel 1985-ben indult, a mai napig olyan cégek használják a mindennapjainkban mint a Google, az Amazon, vagy a Microsoft. A titok a fejlesztésükben rejlik. Elmondható, hogy haladnak a korral, és mindent megtesznek, hogy a felhasználók igényeit ki tudják elégíteni.

De vajon a felhasználók ki tudják használni teljes mértékben az Excel nyújtotta lehetőségeket? A több száz függvény között nehéz eligazodni.

Ehhez nyújtott segítséget az Excel 2020 év végén, hogy leegyszerűsítse a folyamatokat, bevezette a lambda kifejezést.

A lambda kifejezés korábban már jól ismert a Microsoft által használt Visual Studio. S mivel a Visual Studio és az Excel is Microsoft termék egyszerű választás a közös platform a VBA. Viszont új kihívásként, ahogy annak idején az újra számlálást bevezették, fejlesztésre került a lambda is.

Mit jelent az Excelben a lambda függvény?

Írd meg egyszer a képletet, majd használd minden nap. Innentől gyakorlatilag elég egyszer megírni a képletet, elmenteni, egyedi nevet adni neki és már használható is.

Ezzel időt és energiát spórolunk meg, ami hasznot hoz a vállalat számára.

Eddig ez a Visual Studio-ban volt lehetséges.

A Visual Studio többféle programozási nyelvet használ. Többek között a C#, Visual Basic.NET, JavaScript, Python.

Ezt a programozási környezetet szoftver fejlesztésre, webfejlesztésre és játékfejlesztésre használják.
Vonzereje a széles körű funkciókban, a megbízhatóságban, és a bővíthetőségben rejlik. Legnépszerűbb felhasználók közé tartozik, a Microsoft, a Google, az Amazon és a Netflix.

Nézzünk néhány példát a C# és Excel lambda kifejezéseire:

1. példa: átlagszámítás C# és Excel függvényekkel:

5. ábra: Átlagszámítás (Forrás: Saját szerkesztés C#-kód és Excel használatával)

```
List<double> osztalyzat = new List<double> { 90, 95, 100, 85, 92, 88, 91, 89, 94,
    87, 75, 34, 22, 12, 89, 99, 67, 43, 12, 100 };

Console.WriteLine($"Az átlagos osztályzat: {osztalyzat.Average()}");
```

```
Az átlagos osztályzat: 73,2
```

1. Mennyi az átlagos osztályzat?	
numbers	Átagom
90	1
95	2
100	3
85	4
92	5
88	6
91	7
89	8
94	9
87	10
75	11
34	12
22	13
12	14
89	15
99	16
67	17
43	18
12	19
100	20
Eredmény	Eredmény
73,2	11

=Átalgom

=LAMBDA(numbers; ÁTLAG(numbers))(A3:A22)

A legegyszerűbb példa bemutatása. A C# kód előnye, hogy programozható, ezzel lehetővé téve bonyolultabb logikai feltételek beépítését. Hátránya, hogy kódolási ismeretek szükségesek, továbbá nem ad vissza vizuális táblázatot.

Az Excel előnye, hogy jól integrálható további táblázatokban, nem igényelnek programozási ismereteket. Hátránya, hogy korlátozott lehetőségekkel rendelkeznek az programozással szemben és kevésbé automatizálhatók a kódokhoz képest.

A C#-ban a LIST metódus osztályzat-nak elnevezve, meghívja a AVERAGE metódust ezzel kiszámolja a felsorolt számok átlagát.

EXCEL-ben az Átlag függvényt hívom meg a megadott tartomány kiszámítására. A lambda névtelen függvény pedig biztosítja, hogy a képlet újra felhasználható legyen. Tehát nem kell megírnunk bonyolult képleteket, csak egyszer. Ezt a képletet Átlagom-nak neveztem el.

2. példa: 90 fölötti érték számítás C# kóddal és Excelben:

6. ábra: 90 fölötti értékek számítás (forrás: Saját szerkesztés C# kód és Excel használatával)

```
var osztalyzat90folott = osztalyzat.Where(szam => szam > 90);
Console.Write("A 90 fölötti értékek: ");
foreach (var szam in osztalyzat90folott)
{
    Console.Write($"{szam} ");
}
Console.WriteLine();
```

```
A 90 fölötti értékek: 95 100 92 91 94 99 100
```

4. Melyek a 90 feletti értékek?	
	90 felett
Eredmény	Eredmény
95	95
100	100
92	92
91	91
94	94
99	99
100	100

=_feletti90

=LAMBDA(numbers; SZŰRŐ(numbers; numbers>90))(A3:A22)

A var kulcsszóval deklaráljuk a 90 fölötti értékeket. Az osztályzat egy korábbi esetben már definiálva volt. A foreach ciklus segítségével végig megy a program minden egyes szám elemen.

Az Excelben a szűrő funkcióval keressük meg a feltételhez kötött értékeket. A „>90” adja majd a feltételt.

3. példa A Janka nevű emberek listázása C#-ban és Excelben

7. ábra A Janka nevű emberek listázása C#-ban és Excelben (forrás:Saját szerkesztés C# kód és Excel használatával)

```
List<string> nevek = new List<string> { "János", "Julcsi", "József", "Jolán",
    "Jácint", "Jónás", "Jeromos", "Jakab", "Jenő", "Janka", "Judit",
    "Juhász Tamás", "Jónás Elemér", "Tóth János", "Kovács Ferenc" };


var janka = nevek.Where(nevek => nevek.Contains("Janka"));
Console.Write("Az emberek, akinek a neve Janka: ");
foreach (var nev in janka)
{
    Console.Write($"{nev} ");
}
Console.WriteLine();
```

```
Az emberek, akinek a neve Janka: Janka
```

8. Add meg a Janka nevű emberek listáját!			
nevek			
János	János		
Julcsi	Julcsi		
József	József		
Jolán	Jolán		
Jácint	Jácint		
Jónás	Jónás		
Jeromos	Jeromos		
Jakab	Jakab		
Jenő	Jenő		
Janka	Janka		
Judit	Judit		
Juhász Tama	Juhász Tamás		
Jónás Elemé	Jónás Elemér		
Tóth János	Tóth János		
Kovács Fere	Kovács Ferenc		
Eredmény	Eredmény		
Janka	Janka		

=Név_keres

=LAMBDA(nevek;KERES("Janka";F3:F17))(F3:F17)

A var kulcsszóval deklaráljuk a Janka nevet. A where kulcsszóval adjuk meg a kódnak a lambda kifejezést. Foreach-el végig menve az összes adaton kiírja az eredményt a „nev” kód megadására.

Az Excelben a keres függvényt használjuk a Janka szó keresésére.

4. példa J betűvel kezdődő nevek listázása C# és Excelben

8. ábra A J betűvel kezdődő nevek listázása C# és Excelben (Forrás: Saját szerkesztés C# kód és Excel használatával)

```
var nevekJbetuvel = nevek.Where(nev => nev.StartsWith("J"));
Console.Write("Az emberek, akiknek a neve J betűvel kezdődik: ");
foreach (var nev in nevekJbetuvel)
{
    Console.Write($"{nev} ");
}
Console.WriteLine();
```

```
Az emberek, akiknek a neve J betűvel kezdődik: János Julcsi József Jolán Jácint
 Jónás Jeromos Jakab Jenő Janka Judit Juhász Tamás Jónás Elemér
```

10. J betűvel kezdődő nevek	
	J betű
Eredmény	Eredmény
János	János
Julcsi	Julcsi
József	József
Jolán	Jolán
Jácint	Jácint
Jónás	Jónás
Jeromos	Jeromos
Jakab	Jakab
Jenő	Jenő
Janka	Janka
Judit	Judit
Juhász Tama	Juhász Tamás
Jónás Elemé	Jónás Elemér

=J_betű

```
=LAMBDA(nevek;HA(BAL(F3)="J";F3;""))(F3:F17)
```

A var kulcsszóval hagyjuk, hogy a program döntse el, hogy mit szeretne csinálni. Figyelembe veszi, hogy a nevek J betűvel kezdődjenek. Ugyanúgy a nevek definiált listát használom. A where metódussal új lista jön létre, ahol azok a nevek szerepelnek, amelyek J betűvel kezdődnek.

Az Excelben a HA függvény használatával keresem meg a J betűvel kezdődő szavakat. A kezdéshez szükséges megadnom a Bal függvénnyel a kezdés helyét.

5. példa: Mi a szuperhősök igazi neve?

9. ábra Mi a szuperhősök igazi neve? (Forrás: Saját szerkesztés C# és Excel használatával:

```
new SuperHero("Clark Kent", "Superman", new List<string> { "Super Strength", "Flight", "Invulnerability", "Speed", "Heat Vision" }, 35, "Justice League", 100),
new SuperHero("Bruce Wayne", "Batman", new List<string> { "Intelligence", "Peak Human Condition", "Martial Arts", "Stealth" }, 40, "Justice League", 50),
new SuperHero("Diana Prince", "Wonder Woman", new List<string> { "Super Strength", "Speed", "Durability", "Flight" }, 800, "Justice League", 75),
new SuperHero("Barry Allen", "Flash", new List<string> { "Super Speed", "Time Travel", "Intangibility" }, 30, "Justice League", 60),
new SuperHero("Hal Jordan", "Green Lantern", new List<string> { "Energy Constructs", "Flight", "Force Fields" }, 35, "Justice League", 70),
new SuperHero("Arthur Curry", "Aquaman", new List<string> { "Underwater Breathing", "Super Strength", "Telepathy" }, 35, "Justice League", 65),
new SuperHero("Tony Stark", "Iron Man", new List<string> { "Powered Armor", "Genius Level Intellect", "Energy Repulsors" }, 45, "Avengers", 80),
new SuperHero("Steve Rogers", "Captain America", new List<string> { "Super Strength", "Enhanced Agility", "Indestructible Shield" }, 100, "Avengers", 85),
new SuperHero("Thor Odinson", "Thor", new List<string> { "Godly Strength", "Flight", "Weather Control" }, 1500, "Avengers", 90),
```

```
var superHeroNames = superHeroes.Select(hero => new { RealName = hero.RealName, SuperheroName = hero.SuperheroName }).OrderBy(x => x.RealName).ToList();
superHeroNames.ForEach(hero => Console.WriteLine(value:$"{hero.RealName} is {hero.SuperheroName}"));
```

```
Arthur Curry is Aquaman
Barry Allen is Flash
Bruce Wayne is Batman
Clark Kent is Superman
Diana Prince is Wonder Woman
Hal Jordan is Green Lantern
Steve Rogers is Captain America
Thor Odinson is Thor
Tony Stark is Iron Man
```

Valódi név	Hős név
Clark Kent	Superman
Bruce Wayne	Batman
Diana Prince	Wonder woman
Barry Allen	Flash
Hal Jordan	Green Latern
Arthur Curry	Aquaman
Tony Stark	Iron Man
Steve Rogers	Captain America
Thor Odinson	Thor

=Szuperhős_neve

=LAMBDA(valódinév;hősnév;ÖSSZEFŰZ(L20;" igazi neve ";M20))(L20;M20)

Eredmény
Clark Kent igazi neve Superman igazi neve
Bruce Wayne igazi neve Batman igazi neve
Diana Prince igazi neve Wonder woman igazi neve
Barry Allen igazi neve Flash igazi neve
Hal Jordan igazi neve Green Latern igazi neve
Arthur Curry igazi neve Aquaman igazi neve
Tony Stark igazi neve Iron Man igazi neve
Steve Rogers igazi neve Captain America igazi neve
Thor Odinson igazi neve Thor igazi neve

A superHeroes nevű listából kiválasztást végzek, a hero objektumhoz egy új kiválasztást rendelek RealName és SuperheroName oszlopokkal, hogy csak ezt a kettőt mutassa, majd rendezem valódi név szerint és listába teszem a kiválasztott elemeket a ToList() segítségével.

Alatta a ForEach függvénnyel kiíratom konzolra a neveket úgy, hogy a hős valódi neve, majd mellé a szuperhős nevét is.
Excelben először létrehozom az ÖSSZEFŰZ képlettel a kiválasztást, majd, a lambda függvénynek megadok két argumentumot, a képlet végén pedig megadom, hogy honnan válassza ki az adatokat.
A példákon keresztül megállapítható, hogy az Excel és C# közös függvény nevet használ.

6. Következtetések és javaslat

A kutatás eredményei azt mutatják, hogy a lambda kifejezések hatékonyságot adnak az Excel felhasználóinak.

A C# kifejezésekben pedig egyszerűbb, átláthatóbb kódokat nyújtanak a fejlesztőknek. Segítségével nincs szükség professzionális programozási ismeretekre. Amellett pedig képes bonyolultabb rendszerek kezelésére. A neurális hálózatok működésére.

A lambda hozzájárul a további kutatás fejlesztésekhez, a mesterséges intelligencia működéséhez.

A lambda kifejezéseket javaslom nagyméretű Excel-fájlok adatbázisainak feldolgozására. Összetett lekérdezések alkalmazására. Automatizált lekérdezések használatához.

Javaslom a lambda kifejezések elterjesztését szélesebb körben, Excelben.

Oktatási anyagok és oktató videók létrehozásával könnyebbé tenném a felhasználók számára a lekérdezés használatát, annak érdekében, hogy minél jobban kihasználják az Excel nyújtotta lehetőségeket.

Javaslom a lambda kifejezés használatát a C# kódokban, a kódsorok egyszerűsítésére és olvashatóságára vonatkozóan.

Javaslom a lambda kifejezések használatát a neurális hálózatok és a mesteréges intelligencia fejlesztésére.

7. Összefoglalás

A záródolgozatom a lambda és LINQ kifejezésekről írom Excel és C# programokon keresztül.
A lambda kifejezés segíti az Excel automatizálását, hatékonyságát.
A C# Visual Studio programozási környezet és az Excel is Microsoft termék. Ennek köszönhetően, az Excel-ben VBA mentesen lehet lambda kifejezéseket használni. Ez segíti a hatékony, gyors munkavégzést.
Az Excel folyamatos fejlődésével elérte, hogy továbbra is népszerű maradjon. Olyan cégek használják, mint a Microsoft, a Google, vagy az Amazon.
A szakmai gyakorlatomat a Rosenberger Automotive Cabling Kft.-nél végeztem, ahol Excel-ben készítettem makró segítségével formanyomtatványokat.
A Visual Studio és az Excel alkalmazásokban összehasonított képletek bizonyítják, hogy ugyanazt az eredményt el lehet érni egyszerűbben lambda kifejezéssel és ezáltal az Excelben programozás nélkül.
A lambda függvény azért hasznos számunkra, mert a több száz képlet közül nehéz kiválasztani a megfelelőt. De ha már egyszer megalkottunk egy képletet, a rögzítésének segítségével, a képlet újra felhasználható.
A lambda függvény, nem csak az Excel-ben nyújt segítséget, hanem a Visual Studio-ban is. A segítségével a kódok egyszerűbbek, rövidebbek, átláthatóbbak.
A kódsor és a függvény összehasonlításában a közös, hogy mindkét esetben van lambda kifejezés, és a matematikai műveletek szavai megegyeznek.
A lambda hozzájárul a további kutatás fejlesztéshez, a mesterséges intelligencia működéséhez. A nagyméretű Excel fájlok miatt, és az összetett lekérdezések miatt.
Javaslom a lambda kifejezések szélesebb körben való elterjesztését, Excelben.
A lambda kifejezések oktatásával, a Visual Studio-ban a kódok rövidebbek átláthatóbbak, az Excelben rögzíthetőek, ami által csak egyszer kell a függvényt megírni. Továbbá javaslom a lambda kifejezések használatát a neurális hálózatok és a mesterséges intelligencia kifejlesztésére.

Hivatkozások

1. Rosenberger Automotive Cabling honlapja. Letöltés dátuma: 2024.04.16, forrás: https://www.rosenberger.com/company/rosenberger-worldwide/rosenberger-automotive-cabling
2. A Rosenberger hivatalos honlapja. Letöltés dátuma: 2024.04.16, forrás: *https://www.rosenberger.com/company/history/*
3. (Dietrich E.-Samacchia P. (2017): NDepend blog alapján, *C# version history* Letöltés dátuma: 2024.04.16. forrás: *https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history#c-version-10-1*)
4. (Gibson R. 2016-2024 GitHub alapján, *The history of C#* Letöltés dátuma: 2024.04.16, forrás: *https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history#c-version-10-1*)
5. (The history of C#, 2023.12.15. *Overview of LINQ*, Letöltés dátuma: 2024.04.16, forrás: https://learn.microsoft.com/en-us/dotnet/csharp/linq/)
6. (Wagner B.-Klonowski B. *Query expression basics*, 202.03.06. Letöltés dátuma: 2024.04.16, forrás: https://learn.microsoft.com/en-us/dotnet/csharp/linq/)
7. (Kőrösi B. *Az Excel születése – a kiválóság* története!* (2019), Letöltés dátuma: 2024.04.16. forrás: https://sosexcel.com/exceltippek/az-excel-tortenete/)
8. (N. a. *Wikipedia: Microsoft Excel*, Letöltés dátuma: 2024.04.16, forrás: *https://hu.wikipedia.org/wiki/Microsoft_Excel*)
9. (Gordon A.-Jones S. P.(2021)*LAMBDA: The ultimate Excel worksheet function*, Letöltés dátuma: 2024.04.16, forrás: https://www.microsoft.com/en-us/research/blog/lambda-the-ultimatae-excel-worksheet-function/)
10. (N. a. InfoQ, Az Excel képletnyelve immár Turing-teljes, Letöltés dátuma: 2024.04.21, forrás: https://www.infoq.com/articles/excel-lambda-turing-complete/)

Ábrajegyzék

NYILATKOZAT

a záródolgozat nyilvános hozzáféréséről és eredetiségéről

A hallgató neve: TÓTHNÉ ONDRÉK MARIANNA

A Hallgató Neptun kódja: GX10W5

A dolgozat címe: C# LINQ ÉS LAMBDA KIFEJEZÉSEK AZ EXCEL ADATOK LEKÉRDEZÉSEIBEN ÉS MŰVELETEIBEN

A megjelenés éve: 2024

A konzulens intézetének neve: MŰSZAKI INTÉZET

A konzulens tanszékének a neve: ALKALMAZOTT INFORMATIKAI TANSZÉK

Kijelentem, hogy az általam benyújtott záródolgozat[1] egyéni, eredeti jellegű, saját szellemi alkotásom. Azon részeket, melyeket más szerzők munkájából vettem át, egyértelműen megjelöltem, és az irodalomjegyzékben szerepeltettem.

Ha a fenti nyilatkozattal valótlant állítottam, tudomásul veszem, hogy a záróvizsga-bizottság a záróvizsgából kizár és a záróvizsgát csak új dolgozat készítése után tehetek.

A leadott dolgozat, mely PDF dokumentum, szerkesztését nem, megtekintését és nyomtatását engedélyezem.

Tudomásul veszem, hogy az általam készített dolgozatra, mint szellemi alkotás felhasználására, hasznosítására a Magyar Agrár- és Élettudományi Egyetem mindenkori szellemitulajdon-kezelési szabályzatában megfogalmazottak érvényesek.

Tudomásul veszem, hogy dolgozatom elektronikus változata feltöltésre kerül a Magyar Agrár- és Élettudományi Egyetem könyvtári repozitori rendszerébe. Tudomásul veszem, hogy a megvédett és

- nem titkosított dolgozat a védést követően
- titkosításra engedélyezett dolgozat a benyújtásától számított 5 év eltelte után

nyilvánosan elérhető és kereshető lesz az Egyetem könyvári repozitori rendszerében.

Kelt: 2024 év 04 hó 18 nap

Hallgató aláírása

[1] A megfelelő dolgozattípus meghagyása mellett a többi típus törlendő.

NYILATKOZAT

TÓTHNÉ ONDREK MARIANN (név) (hallgató Neptun azonosítója: GX10W5) konzulenseként nyilatkozom arról, hogy a záródolgozatot áttekintettem, a hallgatót az irodalmi források korrekt kezelésének követelményeiről, jogi és etikai szabályairól tájékoztattam.

A záródolgozatot a záróvizsgán történő védésre **javaslom / nem javaslom**[1].

A dolgozat állam- vagy szolgálati titkot tartalmaz: igen nem*[2]

Kelt: GYÖNGYÖS, 2024 év ÁPRILIS hó 16 nap

belső konzulens

[1] A megfelelő aláhúzandó.
[2] A megfelelő aláhúzandó.