СУБОТИЦА
SZABADKA
SUBOTICA
SUBOTICA
2024

13. МЕЂУНАРОДНА МЕТОДИЧКА КОНФЕРЕНЦИЈА

КОМПЕТЕНЦИЈЕ

13. NEMZETKÖZI MÓDSZERTANI KONFERENCIA

KOMPETENCIÁK

13. MEĐUNARODNA METODIČKA KONFERENCIJA

KOMPETENCIJE

13TH INTERNATIONAL METHODOLOGICAL CONFERENCE

COMPETENCES

УНИВЕРЗИТЕТ У НОВОМ САДУ УЧИТЕЉСКИ ФАКУЛТЕТ НА МАЂАРСКОМ НАСТАВНОМ ЈЕЗИКУ У СУБОТИЦИ
ÚJVIDÉKI EGYETEM MAGYAR TANNYELVŰ TANÍTÓKÉPZŐ KAR, SZABADKA
SVEUČILIŠTE U NOVOM SADU UČITELJSKI FAKULTET NA MADARSKOM NASTAVNOM JEZIKU U SUBOTICI
UNIVERSITY OF NOVI SAD HUNGARIAN LANGUAGE TEACHER TRAINING FACULTY, SUBOTICA

13. Међународна методичка конференција

КОМПЕТЕНЦИЈЕ

Зборник радова

Датум одржавања: 7–8. новембар 2024.
Место: Универзитет у Новом Саду, Учитељски факултет на мађарском наставном језику,
Суботица, ул. Штросмајерова 11., Република Србија

13. Nemzetközi Módszertani Konferencia

KOMPETENCIÁK

Tanulmánygyűjtemény

A konferencia időpontja: 2024. november 7–8.
Helyszíne: Újvidéki Egyetem, Magyar Tannyelvű Tanítóképző Kar,
Szabadka, Strossmayer utca 11., Szerb Köztársaság

13. Međunarodna metodička konferencija

KOMPETENCIJE

Zbornik radova

Datum održavanja: 7–8. studeni 2024.
Mjesto: Sveučilište u Novom Sadu, Učiteljski fakultet na mađarskom nastavnom jeziku,
Subotica, ul. Strossmayerova 11., Republika Srbija

13 th International Methodological Conference

COMPETENCES

Papers of Studies

Date: November 7–8, 2024
Address: University of Novi Sad, Hungarian Language Teacher Training Faculty,
Subotica, 11 Štrosmajerova str., Republic of Serbia

Издавач
Универзитет у Новом Саду
Учитељски факултет на мађарском наставном језику
Суботица

Kiadó
Újvidéki Egyetem
Magyar Tannyelvű Tanítóképző Kar
Szabadka

Izdavač
Sveučilište u Novom Sadu
Učiteljski fakultet na mađarskom nastavnom jeziku
Subotica

Publisher
University of Novi Sad
Hungarian Language Teacher Training Faculty
Subotica

Одговорни уредник / Felelős szerkesztő /
Odgovorni urednik / Editor-in-chief

Valéria Pintér Krekić

Уредник / Szerkesztő / Urednik / Editor

Márta Törteli Telek
Éva Vukov Raffai

Технички уредник / Tördelőszerkesztő /
Tehnički urednik / Layout editor

Attila Vinkó
Zsolt Vinkler

+381 (24) 624 444
magister.uns.ac.rs/conf
inter.conf@magister.uns.ac.rs

ISBN 978-86-81960-32-5

Суботица – Szabadka – Subotica – Subotica
2024

САДРЖАЈ
TARTALOM
SADRŽAJ
CONTENTS

TÍPUSFELADATOK MEGOLDÁSI MÓDSZEREI TÁBLÁZATKEZELÉSSEL ÉS PROGRAMOZÁSSAL

TÖRLEY GÁBOR, BERNÁT PÉTER

Eötvös Loránd Tudományegyetem, Informatikai Kar, Budapest, Magyarország
gabor.torley@inf.elte.hu, bernatp@inf.elte.hu

Összefoglaló

A digitális kultúra (informatika) tantárgy egyik kiemelt célja a problémamegoldó gondolkodás fejlesztése. Az adatok tárolásával és feldolgozásával kapcsolatos feladatok a tantárgy egyik jelentős és részletesen feldolgozott témakörét alkotják, amelyek gyakran táblázatkezelő rendszerrel és programozási nyelvvel is megoldhatók. Korábbi cikkeinkben rámutattunk a táblázatkezelés és a programozástanítás összekapcsolásának elvi lehetőségeire. Jelen publikációnkban feltérképezzük a tantárgy jellemző adatfeldolgozási feladatait, problémáit, és tipikus különböző mélységű, részletességű lehetséges alkalmazói és programozási megoldásait, azok fogalmi és módszertani kapcsolódásait, valamint a segítségükkel átadható ismereteket és fejleszthető kompetenciákat.

Kulcsszavak: *táblázatkezelés, programozás, problémamegoldás, algoritmikus gondolkodás, tanítási módszerek*

1. Bevezetés

Amikor a problémamegoldás kompetenciájáról beszélünk, akkor általában a programozás jut először az eszünkbe, mint tevékenység, ami fejleszti az algoritmikus gondolkodást, illetve az absztrakciós készséget. Ugyanakkor a táblázatkezelésről első gondolatként a szoftveralkalmazás területe juthat eszünkbe, illetve második gondolatként a matematika. Amikor a táblázatkezelés és a programozás egy lapon szerepel, akkor a makrók programozása kerül előtérbe, ami inkább programozás, mint táblázatkezelés.

Általában a táblázatkezelést nem sorolják a problémamegoldás eszközei közé, holott a hatályos kerettanterv (Oktatási Hivatal, 2020a) a problémamegoldás fejlesztésének egy eszközeként nevezi meg azt, hasonlóan a programozáshoz.

Korábbi cikkeinkben (Törley, Zsakó és Bernát, 2022; Törley és Bernát, 2021) elméleti szinten hasonlítottuk össze a táblázatkezelés és a programozás fogalomrendszerét, valamint, hogy miként lehet alkalmazni, vizualizálni a programozás egyes sémaalgoritmusait táblázatkezelőben.

Ebben a publikációban az esettanulmány módszerét használva mutatjuk meg a táblázatkezelés és a programozás közötti didaktikai kapcsolatot, feladattípusokon, valamint azok különböző mélységű, részletességű megoldásain keresztül. Minden esetben először a táblázatkezelőben, majd pedig a programozással elvégezhető megoldást tárgyaljuk, mert a közoktatásban a táblázatkezelés mindig megelőzi a programozást a tanított témák sorrendjében.

A konkrét megoldásokat a táblázatkezelésen és a programozáson belül is egy-egy konkrét eszközzel mutatjuk be: a táblázatkezeléshez a hazai közoktatásban leggyakrabban használt rendszert, a Microsoft Excelt, a programozáshoz pedig a C# programozási nyelvet választottuk, mert az érettségin is lehet használni ezt a nyelvet, és mert az ELTE programtervező informatikus és informatikatanár képzésein ezt a nyelvet használjuk a programozásba bevezető kurzusok során. A programkódok mellett azok pszeudokódban írt algoritmusait is megadjuk.

2. Irodalmi áttekintés

Szalayné szerint (2016) a táblázat egy programnak tekinthető adatokkal és előre meghatározott algoritmusokkal. Bár a tanulók egy táblázatot/táblázatkezelőt látnak, mégis meg kell érteniük az általuk írt „programot”, azaz a függvényekkel megvalósított megoldásukat. Ilyen módon táblázatkezelő alkalmazásával lehet indirekt módon programozást tanítani.

Bíró és Csernoch szerint (2015) a táblázatkezelő szoftver használható a problémamegoldás eszközeként, és az általuk leírt Sprego módszer alkalmas a tanulók számítógépes gondolkodásának és algoritmikus készségének fejlesztésére.

Több programozási alapfogalomnak megvan a „párja” a táblázatkezelés fogalomtárában. Kankuzi és kollégái azt javasolják (2017), hogy a programozás bevezetése előtt érdemes táblázatkezelést tanítani, amely során a programozás alapfogalmait indirekt módon lehet bevezetni, a táblázatkezelés által megoldott problémákon keresztül.

Warren szerint (2004), ha először táblázatkezelőt használunk, és utána egy választott programozási nyelvet, akkor gyorsabban el lehet jutni a tananyagban a bonyolultabb algoritmusokig.

3. Feladattípusok

A közoktatásban a táblázatkezelés és a programozás témakörén belül is jellemzően adatsokaságokkal kapcsolatos feladatokat kell megoldani. A táblázatkezelésben ez az adatsokaság szinte mindig egy (vagy több) táblázat, a programozástanítás során azonban a többdimenziós adatszerkezetekkel kapcsolatos feladatokat megelőzik az egydimenziós adatszerkezetekre (például tömbökre) vonatkozók. Ugyanakkor mindkét témakör tanítására jellemző a feladatok típusfeladatokba történő csoportosítása a megoldási módszereik alapján.

A példa kedvéért válasszunk adatsokaságnak egy olyan táblázatot, amely egy elképzelt országos komplex természettudományi verseny eredményeit foglalja össze. Ezen a versenyen 11. és 12. tanulók indulhattak, és matematikából, fizikából és informatikából is legfeljebb 100-100 pontot gyűjthettek. Az ötletet a Zalaegerszegi Zrínyi Miklós Gimnázium által 1992 óta évente megrendezett Izsák Imre Gyula Természettudományi Versenyből merítettük.) A táblázat tartalmazza a résztvevő versenyzők nevét, az iskolájuk vármegyéjét, az évfolyamukat, valamint az említett három tantárgyból elért pontszámukat (1. táblázat).

1. táblázat: *A cikkben vizsgált feladat alapjául szolgáló adatok*

Név	Vármegye	Évfolyam	Matematika	Fizika	Informatika
Bodnár Anna	Zala	12	22	15	71
Hamrik Szabina	Budapest	12	4	40	21
Hegedüs Péter	Zala	11	75	73	81
Kenéz Balázs	Pest	12	3	0	37
Lakatos Márton	Somogy	12	97	43	83
Marosi Tóbiás	Budapest	11	53	63	100
Tallósi Mónika	Budapest	11	20	20	24
Tóth Viktor	Somogy	12	100	68	100
Törő Norbert	Budapest	12	8	53	51
Vavrik Márton	Pest	11	24	30	33

Ezzel az adatsokasággal kapcsolatban például a következő típusfeladatok és konkrét feladatok tűzhetők ki (2. táblázat).

2. táblázat: *Feladattípusok és egy-egy konkrét példa*

Típus	Feladat
Rendezés	Rendezzünk név szerint!
Összesítés	Összesen hány pontot értek el informatikából a versenyzők?
Kiválogatás	Válogassuk ki a fővárosi versenyzőket!
Feltételes összesítés	Összesen hány pontot értek el informatikából a fővárosi versenyzők?
Megszámolás	Hány fővárosi versenyző indult a versenyen?
Eldöntés	Indult-e fővárosi tanuló a versenyen?
Kiválasztás, keresés	Hány pontot ért el informatikából Horváth Anna?

A feltételes összesítés feladattípusába tartozónak tekintjük a feltételes összeadás mellett (amelyet szűkebben feltételes összegzésnek hívunk) a feltételes átlagszámításra és a feltételes minimum- és maximumkiválasztásra vonatkozó feladatokat is.

A programozásban a felsorolt feladattípusokhoz típusalgoritmusokat, illetve azok kombinálását lehet rendelni. A NAT 2022 alapján készült Digitális kultúra 10. tankönyv (Oktatási Hivatal, 2022b) is megnevez típusalgoritmusokat: sorozatszámítás (összegzés és átlagolás), eldöntés, kiválasztás, keresés, megszámolás, maximum- és minimumkiválasztás. Az egyetemi szintű oktatásban, a programozás bevezető kurzusában programozási tételeknek nevezzük ezeket a típusalgoritmusokat. Ezek feladatosztályoknak, feladatcsoportoknak tekinthetőek. Ez a felosztás azért hasznos, mert ha egy feladattípus megoldási módját megértjük, akkor az összes, adott osztályba tartozó feladatot meg tudjuk oldani. Ebből következik, hogy a konkrét feladatot mindig vissza tudjuk vezetni valamelyik feladattípushoz/sémához. További előnye ezeknek a típusfeladatoknak, hogy helyességük matematikailag bizonyítható, ezért nevezzük őket programozási tételeknek Egyetemi szinten még két feladattípust adunk hozzá a fenti hathoz: másolás (függvényszámítás) és kiválogatás. (Szlávi, Törley és Zsakó, 2019).

4. A feltételes összesítés és altípusai

Azt, hogy a táblázatkezelésben és a programozásban is különböző mélységű, kifejtettségű megoldások adhatók ugyanarra a problémára, valamint, hogy a különböző megoldások között milyen didaktikai kapcsolatok találhatók és használhatók ki, a feltételes összesítés típusfeladatán keresztül fogjuk bemutatni. Egy-egy típusfeladaton belül különböző altípusok is meghatározhatók. Például a feltételes összesítés típusába tartozó feladatokat megkülönböztethetjük elsősorban a feltétel típusa alapján, másodsorban pedig aszerint, hogy az összesítendő adatok el vannak tárolva, vagy azokat ki kell számítani (3. táblázat).

3. táblázat: *Feladat-altípusok és egy-egy példa*

Altípus	Példafeladat
Egyszerű feltétel	Összesen hány pontot értek el informatikából a *fővárosi* versenyzők?
ÉS-kapcsolatos feltétel	Átlagosan hány pontot értek el informatikából a *fővárosi 11.-es* versenyzők?
VAGY-kapcsolatos feltétel	Átlagosan hány pontot értek el informatikából azok, akik *valamely tantárgyból legalább 80 pontot gyűjtöttek*?
Számított feltétel	Mi a legkisebb előforduló informatika-pontszám azon versenyzők között, akik *a három tárgyból együtt legalább 200 pontot szereztek?*
Számított értékek összesítése	Átlagosan hány pontot értek el *a három tárgyból együtt* a fővárosi versenyzők?

Vizsgálatunk tárgyát a feltételes összesítés típusfeladatán belül elsősorban az első altípus képezi majd, vagyis arra a kérdésre fogunk sokféleképpen választ adni, hogy *összesen hány pontot értek el informatikából a fővárosi versenyzők.* Ugyanakkor ki fogunk térni arra is, hogy a megoldások menetében milyen lényegesebb különbségekre lehet számítani a többi altípus esetén.

5. Feltételes összesítés beépített függvénnyel

5.1 Táblázatkezelés

A táblázatkezelő rendszerek beépített függvényeket (is) kínálnak a különböző feltételes összesítések elvégzésére. Az Excelben a feltétel típusán túl az összesítés jellegét (például összeg, átlag vagy szélsőérték) is figyelembe kell venni a megfelelő függvény kiválasztásakor. Az *„Összesen hány pontot értek el informatikából a fővárosi versenyzők?”* kérdésben egy egyszerű (nem összetett) feltétel található (csak a budapesti versenyzőkkel kapcsolatban kívánunk összegezni), az összesítés jellege pedig az összeadás. Ebben az esetben a SZUMHA függvénnyel válaszolható meg a kérdés (1. ábra).

	A	B	C	D	E	F	
1	**Név**	**Megye**	**Évf.**	**Mat.**	**Fiz.**	**Inf.**	
2	Bodnár Anna	Zala	12	22	15	71	
3	Hamrik Szabina	Budapest	12	4	40	21	
4	Hegedüs Péter	Zala	11	75	73	81	
…	…	…	…	…	…	…	
11	Vavrik Márton	Pest	11	24	30	33	
12							
13						196	`=SZUMHA(B2:B11;"Budapest";F2:F11)`

1. ábra: *A feladat megoldása beépített függvénnyel*

Ha ugyanezen versenyzők informatika-pontszámának az átlagára, minimumára vagy maximumára lettünk volna kíváncsiak, akkor az ÁTLAGHA, MINHA, MAXHA függvényeket használhattuk volna. Egy másik, csak ÉS-kapcsolatokat tartalmazó összetett feltétel esetén a SZUMHATÖBB, ÁTLAGHATÖBB, MINHA és MAXHA függvények közül, egy vagy több VAGY-kapcsolatot is tartalmazó feltétel esetén pedig az AB.SZUM, AB.ÁTLAG, AB.MIN és AB.MAX függvények (úgynevezett adatbázisfüggvények) közül választhatnánk az összesítés jellegétől függően. Számított feltételt szintén az adatbázisfüggvényekkel tudunk megfogalmazni. A számított értékek összesítésére azonban ennek a függvénycsaládnak a tagjai sem képesek, helyettük tömbképlet írható. A feltételes összesítés többi, korábbi altípusa esetén azonban elmondható, hogy egyetlen kész eszköz (függvény) használatával megválaszolhatók.

5.2 Programozás

A programozás esetén, a C# nyelvben nem található olyan beépített eszköz (legalábbis a bevezető kurzusokban nem fordul elő), amely egy az egyben megvalósítaná az Excel feltételes összesítő függvényeit. Ugyanakkor, célszerű lehet hivatkozni ezekre a feltételes összesítő függvényekre a konkrét programozási cél megértésének az elősegítésére.

6. Feltételes összesítés kiválogatással és összesítéssel

Az alapkérdésünkre korábban adott direkt megoldást két, egymás után elvégzendő lépésre bonthatjuk: először válogassuk ki a versenyzők közül a fővárosiakat, majd az informatika-pontszámok közül csak az ezen versenyzőkhöz tartozókat adjuk össze!

6.1 Táblázatkezelés

A két lépés egy szemléletes (de a kiindulási adatok megváltozásakor újra végrehajtandó) megvalósítása az Excelben a szűrő funkció és a gyorskalkuláció egymás utáni használata. A szűrővel először megjeleníthetjük csak a budapesti versenyzők sorait, majd az informatika-pontszámokat tartalmazó oszlopban a hozzájuk tartozó pontszámok kijelölését követően az Excel-ablak állapotsorában leolvashatjuk (többek között) a kijelölt adatok összegét (2. ábra).

	A	B	C	D	E	F
1	**Név**	**Megye**	**Évfolyam**	**Mat.**	**Fiz.**	**Inf.**
3	Hamrik Szabina	Budapest	12	4	40	21
7	Marosi Tóbiás	Budapest	11	53	63	100
8	Tallósi Mónika	Budapest	11	20	20	24
10	Törő Norbert	Budapest	12	8	53	51
	Átlag: 49	Cellák száma: 4		Összeg: 196		

2. ábra: *A feladat megoldása szűréssel és gyorskalkulációval*

Ugyanennek a két lépésnek az adatkövető (tehát a kiindulási adatok megváltozásakor automatikusan frissülő) kivitelezése a SZŰRŐ és a SZUM függvény egymás utáni használata. Először a SZŰRŐ függvénnyel megkaphatjuk a fővárosi versenyzők informatika-pontszámát (a szemléltetés kedvéért ezt a munkalap egy üres területén külön is elvégezhetjük), majd a kapott adattömböt a SZUM függvény segítségével összegezhetjük (3. ábra).

	A	B	C	D	E	F	G	H	
1	**Név**	**Megye**	**Évf.**	**Mat.**	**Fiz.**	**Inf.**		**Szűrés**	
2	Bodnár…	Zala	12	22	15	71		21	=SZŰRŐ(F2:F11;B2:B11="Budapest")
3	Hamrik…	Budapest	12	4	40	21		100	
4	Hegedüs …	Zala	11	75	73	81		24	
5	Kenéz…	Pest	12	3	0	37		51	
…	…	…	…	…	…	…			
11	Vavrik…	Pest	11	24	30	33			
12									
13						196	=SZUM(SZŰRŐ(F2:F11;B2:B11="Budapest"))		

3. ábra: *A feladat megoldása szűréssel és összegzéssel*

A szűrő funkcióra és a gyorskalkulációra épülő szemléletes megoldás bonyolódik, ha az összetett feltétel vagy VAGY-kapcsolatokat, vagy számított feltételt is tartalmaz. Ekkor ugyanis az egyszerű helyett irányított szűrésre van szükség, amely során az adatbázisfüggvények kritériumtartományához hasonló szűrőtartomány használható a bonyolult feltételrendszer megfogalmazásához.

A SZŰRŐ és a SZUM függvényt alkalmazó adatkövető megoldás azonban az imént említett különlegesebb feltételektől sem nem lesz lényegesen bonyolultabb: a SZŰRŐ függvényben a matematikai és a logikai műveletekkel, valamint a zárójelezéssel sokféle feltétel képezhető.

6.2 Programozás

A programozás témakörén belül az algoritmus létrehozását mindig meg kell, hogy előzze az adatszerkezet megtervezése és elkészítése. Ugyanis, a táblázatkezeléssel ellentétben, nem áll a rendelkezésünkre olyan összetett adattípus, amely egyrészt megvalósítja a táblázat összes cellájára való hivatkozást, másrészt pedig kifejezi a táblázat sorainak az összetartozását. Az előbbit a mátrix, mint adatszerkezet ugyan teljesítené, de a soron belüli elemeinek az összetartozását már nem fejezné ki: például egy szöveg típusú érték esetén nem derülne ki, hogy az az adott versenyző nevét jelenti. Egy további probléma a mátrixszal, hogy definíció szerint az elemeinek ugyanolyan típusúnak kellene lenniük. Emiatt a programozás során a tömb adatszerkezetet fogjuk használni, ahol a tömb elemei rekordok. Így az elemek (lényegében a sorok) indexelése megmarad, és a rekord mezőivel az egy „sorban” levő elemek összetartozását ki fogjuk tudni fejezni. Megjegyezzük, hogy a bevezető kurzusokon nem a rekord típussal vezetjük be a sémaalgoritmusokat, viszont a jelen tanulmányban leírt alapelvek ugyanilyen módon működnek pl. egész számokból álló tömbökre is.

```
struct Versenyző
{
    public string név;
    public string vármegye;
    public int évfolyam;
    public int matematika;
    public int fizika;
    public int informatika;
}
Versenyző[] versenyzők = new Versenyző[vdb];
```

4. ábra: *A bemeneti adatok adatszerkezete*

A 4. ábrán látható a táblázatnak megfelelő adatszerkezet, illetve a tömb deklarációja, ahol a vdb változó jelenti a versenyzők darabszámát.

A C# nyelvben az aggregátum-függvények, a lambda kalkulus, valamint a LINQ ad eszközöket a kezünkbe ahhoz, hogy a nyelv beépített függvényeit használhassuk a feladat megoldása során. Több más programozási nyelven is rendelkezésünkre állnak hasonló vagy ugyanilyen eszközök.
Beépített függvényekkel az alábbi módon tudjuk megoldani a feladatot (5. ábra).

```
int bp_összeg_inf = versenyzők.Where(elem => elem.vármegye ==
"Budapest").Select(elem => elem.informatika).Sum();
```

5. ábra: *Megoldás C# nyelven, beépített eszközökkel*

Where-rel kiválogatjuk a budapesti versenyzőket, a szűrt eredményből kiválasztjuk az „Informatika” oszlopot (Select), és ebből az „oszlopból” összeget számolunk (Sum()). A gondolkodásmód teljes mértékben megegyezik a táblázatkezelésben látottakkal: kiszűrjük a megfelelő elemeket és összegezzük azokat. Az egyetlen különbséget a két eszközben használt adatszerkezetek különbsége adja: a programozás esetén külön ki kellett választanunk a megfelelő rekordmezőt, azaz az „oszlopot”.

7. Feltételes összesítés kiválogatással és összesítéssel, algoritmikusan

Az előbb ismertetett kétlépéses megoldást tovább finomíthatjuk.

7.1 Táblázatkezelés

A táblázatkezelés tanítása során a feltételesen összegző függvények bevezetésekor (SZUMHA, ÁTLAGHA, MINHA, MAXHA) érdemes a tanulókkal közösen megvizsgálni, hogy a korábban már megismert függvényekkel (és segédcellák alkalmazásával) képesek vagyunk-e feltételes összesítést végezni. Például, a vizsgálatunk fő tárgyát képező kérdés esetén, a HA és a SZUM függvények ismeretében, a következő ötlet körvonalazódhat. Először töltsünk ki HA függvénnyel egy segédoszlopot a fő táblázattól jobbra úgy, hogy a fővárosi versenyzők sorába az elért informatika-pontszámukat írjuk, a többiek esetén azonban „semmit” (de legalábbis az összeadás szempontjából neutrális adatot, például üres szöveget)! Másodszor pedig végezzünk összeadást ebben az oszlopban (6. ábra)!

	A	B	C	D	E	F	G	H	
1	**Név**	**Megye**	**Évf.**	**Mat.**	**Fiz.**	**Inf.**		**Szűrés**	
2	Bodnár Anna	Zala	12	22	15	71			=HA(B2="Budapest";F2;"")
3	Hamrik Szabina	Budapest	12	4	40	21		21	=HA(B3="Budapest";F3;"")
4	Hegedüs Péter	Zala	11	75	73	81			=HA(B4="Budapest";F4;"")
5	Kenéz Balázs	Pest	12	3	0	37			=HA(B5="Budapest";F5;"")
6	Lakatos Márton	Somogy	12	97	43	83			=HA(B6="Budapest";F6;"")
7	Marosi Tóbiás	Budapest	11	53	63	100		100	=HA(B7="Budapest";F7;"")
8	Tallósi Mónika	Budapest	11	20	20	24		24	=HA(B8="Budapest";F8;"")
9	Tóth Viktor	Somogy	12	100	68	100			=HA(B9="Budapest";F9;"")
10	Törő Norbert	Budapest	12	8	53	51		51	=HA(B10="Budapest";F10;"")
11	Vavrik Márton	Pest	11	24	30	33			=HA(B11="Budapest";F11;"")
12									
13								196	=SZUM(H2:H11)

6. ábra: *Megoldás kiválogatással és összegzéssel, algoritmikusan*

Ebben a megoldásban a kiválogatást (szűrést) algoritmikusan végeztük: a táblázatot sorról sorra bejártuk, és kiválogattuk a megfelelő tanulókat, pontosabban az általuk elért pontszámokat. Az összeadást azonban a már ismertnek tekintett SZUM függvénnyel végeztük, a táblázatkezelés témakörében ugyanis nem motivált a SZUM függvény (és a többi elemi összesítőfüggvény) tevékenységének alacsonyabb szintű megvalósítása.

7.2 Programozás

Ahogyan az alfejezet címe is utal rá, ennél a feladatnál két programozási tételt kombinálunk. A feladat tipikus példája annak, ahogyan két sémaalgoritmust egymás után végre lehet hajtani, így alkalmas bevezető példaként feladattípusok kombinálásához.

Az alábbi algoritmus és kód bemutatja, hogy nagyon hasonlóan gondolkodhatunk, mint a táblázatkezelés során, a 6. ábra megoldásánál. A különbség csak annyi, hogy átugorjuk a „nem kellő elemeket”, és nem tárolunk el helyettük üres értékeket.

Algoritmus	Kód
Változó i: Egész	

db := 0
Ciklus i=1-től vdb-ig
  Ha versenyzők[i].vármegye = ”Budapest” akkor
  db := db + 1
  bp_inf[db] := versenyzők[i].informatika
  Elágazás vége
Ciklus vége
	int db = 0;
	for (int i = 0; i <= vdb - 1; i++)
	{
	  if (versenyzők[i].vármegye == "Budapest")
	  {
	  db = db + 1
	  bp_inf[db-1] = versenyzők[i].informatika;
	  }
	}

összeg := 0
Ciklus i=1-től db-ig
  összeg := összeg + bp_inf[i]
Ciklus vége
	int összeg = 0;
	for (int i = 0; i <= db - 1; i++)
	{
	  összeg = összeg + bp_inf[i];
	}

7. ábra: *Kiválogatás és összegzés – egymás után*

A 7. ábrán piros keretben szerepel a kiválogatás, kékben pedig az összegzés algoritmusa és kódja. Könnyen belátható, hogy nem hatékony még az algoritmus, a hatékonyabbá tételt a következő alfejezetben tárgyaljuk. A fenti példa azt kívánta bemutatni, hogy természetesen merül fel először a „két lépéses” gondolkodás, hiszen a budapestiek informatika pontszámát szeretnénk összegezni, és ehhez először szükségünk van a budapestiek pontszámaira.

A feltételes átlagszámítás, maximum- és minimumkeresésnél a piros keretben levő kiválogatás mindig ugyanaz, tehát a gondolkodásmód megegyezik a táblázatkezelésnél leírtakkal. A feltétel teljesülését a db változó értéke fogja minden esetben megmutatni: ha a db változó értéke nagyobb, mint 0, akkor lehet átlagot számolni (az összeg változót elosztjuk a db változó értékével), illetve feltételes maximum- és minimumkeresés esetén alkalmazzuk a maximum/minimumkiválasztás algoritmusát.

8. Feltételes összesítés algoritmikusan

A legtöbb lépésre bontott algoritmikus megoldásunkat a fejben számolás módszere ihlette. Ha a táblázat csak nyomtatva állna a rendelkezésünkre, és úgy kellene fejben kiszámolnunk a fővárosi tanulók informatika-pontszámainak az összegét, akkor a következő algoritmust követhetnénk: haladjunk egyesével a „Vármegye” oszlop adatain felülről lefelé, és minden alkalommal, amikor a „Budapest” adatot találjuk a cellában, akkor adjuk hozzá az ebben a sorban található informatika-pontszámot a fejben folyamatosan számon tartott részösszeghez (és ez a részösszeg kezdetben legyen 0). Miután a „Megye” oszlop utolsó celláját is feldolgoztuk, a számon tartott részösszeg lesz egyúttal a végső összeg. Ez az algoritmikus megoldás az előző pontban ismertetett megoldással szemben nem állítja elő külön a fővárosi versenyzők informatika-pontszámait tartalmazó adatszerkezetet, hanem a szükséges informatika-pontszámokat rögtön egy részösszeghez hozzáadja.

8.1 Táblázatkezelés

Az ismertetett megoldást az Excel lehetőségeivel a következőképpen állíthatjuk elő (8. ábra).

	A	B	C	D	E	F	G	H	
1								**Részö.**	
2	**Név**	**Megye**	**Évf.**	**Mat.**	**Fiz.**	**Inf.**		0	
3	Bodnár …	Zala	12	22	15	71		0	=HA(B3="Budapest";H2+F3;H2)
4	Hamrik …	Budapest	12	4	40	21		21	=HA(B4="Budapest";H3+F4;H3)
5	Hegedüs …	Zala	11	75	73	81		21	=HA(B5="Budapest";H4+F5;H4)
6	Kenéz …	Pest	12	3	0	37		21	=HA(B6="Budapest";H5+F6;H5)
7	Lakatos …	Somogy	12	97	43	83		21	=HA(B7="Budapest";H6+F7;H6)
8	Marozsák …	Budapest	11	53	63	100		121	=HA(B8="Budapest";H7+F8;H7)
9	Tallósi …	Budapest	11	20	20	24		145	=HA(B9="Budapest";H8+F9;H8)
10	Tóth …	Somogy	12	100	68	100		145	=HA(B10="Budapest";H9+F10;H9)
11	Törő …	Budapest	12	8	53	51		196	=HA(B11="Budapest";H10+F11;H10)
12	Vavrik …	Pest	11	24	30	33		196	=HA(B12="Budapest";H11+F12;H11)

8. ábra: *Algoritmikus megoldás táblázatkezelőben*

8.2 Programozás

Ha létezne olyan programozási környezet, amelyben táblázatos formában követhetnénk nyomon az adataink alakulását a program futása közben, akkor valami hasonlót kapnánk (a cellák háttérszínezését kivéve) (9. ábra).

versenyzők[i]							**i**	**összeg**
Név	**Vármegye**	**Évf.**	**Mat.**	**Fiz.**	**Inf.**			0
Bodnár Anna	Zala	12	22	15	71		1	0
Hamrik Szabina	Budapest	12	4	40	21		2	21
Hegedüs Péter	Zala	11	75	73	81		3	21
Kenéz Balázs	Pest	12	3	0	37		4	21
Lakatos Márton	Somogy	12	97	43	83		5	21
Marosi Tóbiás	Budapest	11	53	63	100		6	121
Tallósi Mónika	Budapest	11	20	20	24		7	145
Tóth Viktor	Somogy	12	100	68	100		8	145
Törő Norbert	Budapest	12	8	53	51		9	196
Vavrik Márton	Pest	11	24	30	33		10	196

9. ábra: *A feladat megoldásának táblázatos „nyomon követése”*

A megoldás algoritmusa és kódja a 10. ábrán látható.

Algoritmus	**Kód**

Változó
  i: Egész
összeg := 0
Ciklus i=1-től vdb-ig
  Ha versenyzők[i].vármegye = ”Budapest” akkor
  összeg := összeg + versenyzők[i].informatika
Ciklus vége
	int összeg = 0;
	for (int i = 0; i <= vdb - 1; i++)
	{
	  if (versenyzők[i].vármegye == "Budapest")
	  {
	  összeg = összeg + versenyzők[i].informatika;
	  }
	}

10. ábra: *Feltételes összegzés algoritmusa és kódja*

Az előző megoldásból programtranszformációk után jutunk el ide (10. ábra). (Így is tanítjuk az egyetemen a tanárszakosokat és a kezdő programozókat). Ebben az esetben egy az egyben ugyanúgy gondolkodunk, mint a táblázatkezelésnél, ráadásul a 9. ábrát összevetve a fenti ábrával, látható, hogy a ciklus i. lépésében az összeg változó értéke meg fog egyezni a táblázatban látható értékkel. A sárgával jelzett soroknál fog az algoritmus belépni az elágazás igaz ágába, és akkor fog változni az összeg változó értéke. A két típusalgoritmust egy ciklusba szerveztük, ezért programozás órán feltételes összegzésként szoktunk rá hivatkozni, de az algoritmust megvizsgálva észrevehetőek mind a kiválogatás, mint az összegzés jegyei.

A 8. ábrán, a HA függvény szerepe megegyezik az algoritmusban lévő elágazás vezérlési szerkezetével. Ha a tömb aktuális eleme teljesíti a feltételt, akkor hozzáadjuk az eddigi összeghez az aktuális elem értékét, különben nem adjuk hozzá (azaz semmi nem történik). Átlagszámításnál az elágazás igaz ágában számolnunk kell a feltételt teljesítő elemek darabszámát (ezt a változót inicializálnunk kell a kiválogatáshoz hasonlóan), majd, ha a darabszám pozitív, akkor el tudjuk végezni az átlagszámításhoz szükséges osztást.

A feltételes maximum- és minimumkeresésnél nincs szükség a megfelelő elemek darabszámának az eltárolására. Ilyenkor egyrészt szükség lesz egy logikai típusú változó, amelyből az algoritmus végén ki lehet deríteni, hogy volt-e a feltételnek megfelelő elem a bemeneti sorozatban, másrészt szükségünk lesz arra, hogy egy megfelelő értékkel inicializáljuk azt a változót, amely az algoritmus végén tárolni fogja a feltételnek megfelelő legnagyobb/legkisebb értéket. Ez a feltételes maximumnál a -∞, a feltételes minimumnál a +∞. A számlálós ciklusban lévő elágazás feltétele bővül. Nem akkor lépünk az igaz ágra, ha fővárosi az illető diák, hanem akkor, ha e mellett nagyobb (minimumkeresésnél kisebb) értéket vizsgálunk, mint az eddigi legnagyobb (legkisebb). A feltétel teljesülése esetén cseréljük a korábbi szélsőértéket a jelenlegivel. Belátható, hogy a kezdőérték választása miatt a sorrendben első megfelelő értéknél mindenképp az elágazás igaz ágára fog lépni az algoritmus.

Összetett logikai állításokat tartalmazó feladatok megoldásánál nem kell új eszközökhöz nyúlnunk a programozás tanítása során, elég, ha a logikai kifejezésben a megfelelő helyeken használjuk az ÉS és a VAGY műveleteket. A következő lépés ezen a területen az lesz, amikor az elágazás feltétele valamilyen logikai függvény, ami a bemeneti tömbről vagy egyes elemeiről fog kiértékelni egy állítást. Táblázatkezelésnél ilyenkor jutunk el a tömbfüggvényekhez, de ez a tananyag kizárólag a tehetséggondozáson, versenyfelkészítésen és az egyetemi tanárképzésen kerül elő.

9. Összefoglalás, konklúzió

A fentiek alapján egyrészt beláthatjuk, hogy a táblázatkezelést (a táblázatformázás és a diagramformázás kivételével) miért sorolják inkább a számítástudományi ismeretekhez (az algoritmizálással, programozással együtt), mint a digitális írástudáshoz. Látható, hogy jelentős részben ugyanarról a problémamegoldási területről szól, mint az algoritmizálás és a programozás.

Tanulmányunkban megmutattuk azt is, hogy a táblázatkezelés és a programozás feladattípusai között nagy hasonlóságot találhatunk, ami előrevetíti, hogy egymás segítségére lehetnek a fogalmaik tanulásában. Ráadásul az előbbi állítás a feladattípusok szintjeire is igaz.

A programozástanulást lehet a táblázatkezelési ismeretekre építeni (és viszont), mert a táblázatkezelőt tekinthetjük egy olyan vizualizációs eszköznek, amely egyszerre képes megmutatni a program állapotterét (bemenetet, kimenetet, segédváltozók értékeit), valamint vizualizálni a feladat megoldásához elkészült algoritmust. Ebből következik, hogy a két terület együtt tudja fejleszteni a gondolkodási műveleteket: többek között az analízist, a szintézist, az összehasonlítást, az analógiát és az absztrakciót.

Klasszikus tanítási sorrendben előbb tanulnak a diákok táblázatkezelést, mint típusalgoritmusokat (programozási tételeket), azaz a programozástanítást lehetne a táblázatkezelési ismeretekre építeni és/vagy a programozás bevezetése közben táblázatkezelőt használni.

IRODALOMJEGYZÉK

Biró, Piroska, Csernoch, Mária (2015): Algoritmusok és/vagy táblázatkezelés? In: Ujhelyi, Adrienn és Lévai, Dóra (szerk.): *VII. OktatásInformatikai Konferencia Tanulmánykötet 2015* Budapest: ELTE Pedagógiai és Pszichológiai Kar, 97–111.

Kankuzi, Bennett, Isong, Bassey és Letlonkane, Lucia (2017): Using the spreadsheet paradigm to introduce fundamental concepts of programming to novices. In: Janet, Liebenberg (szerk.): *Proceedings of SACLA'17*, Potchefstroom: North-West University, 39–45.

Oktatási Hivatal (2020a): *A 2020-as NAT-hoz illeszkedő tartalmi szabályozók* Forrás: https://www.oktatas.hu/kozneveles/kerettantervek/2020_nat (2024. 11. 30.)

Oktatási Hivatal (2020b): *Digitális kultúra 10. tankönyv* Forrás: https://www.tankonyvkatalogus.hu/storage/pdf/OH-DIG10TA__teljes.pdf (2024. 11. 30.)

Szalayné Tahy, Zs. (2016): How To Teach Programming Indirectly – Using Spreadsheet Application. *Acta Didactica Napocensia*, 9. 1. 15-22.

Szlávi, P., Törley, G. és Zsakó, L. (2019): Programming theorems have the same origin. *Central-European Journal of New Technologies in Research, Education and Practice*. 1. 1. 1–12.

Törley, Gábor és Bernát, Péter (2021): Spreadsheet As An Algorithm Visualization Tool. In: Abonyi-Tóth, Andor, Stoffa, Veronika és Zsakó, László (szerk.): *XXIV. DidMatTech 2021*. Budapest: ELTE Faculty of Informatics, 15–27.

Törley, G., Zsakó, L. és Bernát, P. (2022): Didactic Connection between Spreadsheet and Teaching Programming. *Athens Journal of Technology and Engineering*. 9. 2. 77-94.

Warren, Peter (2004): Learning to program: spreadsheets, scripting and HCI. In: Lister, Raymond és Young, Alison (szerk.): *Proceedings of the Sixth Australasian Conference on Computing Education – vol. 30*, Darlinghurst, Australian Computer Society, Inc. 327–333.

METHODS FOR SOLVING TYPICAL TASKS IN SPREADSHEET AND PROGRAMMING

Abstract

One of the primary goals of the informatics (computation) subject is to develop problem-solving thinking. Tasks related to data storage and processing constitute a significant and thoroughly covered topic within the subject, and they are often solvable using both spreadsheet software and programming languages. In our previous articles, we highlighted the theoretical possibilities of linking teaching spreadsheet and programming. In this publication, we explore the various possible solutions in spreadsheet and in programming of typical data processing tasks and problems, and also their conceptual and methodological connections, as well as the knowledge and competencies that can be imparted and developed with their help.

Keywords: *spreadsheet, programming, problem solving, algorithmic thinking, teaching methodologies*

CIP - Каталогизација у публикацији
Библиотеке Матице српске, Нови Сад

371.13(082)
371.3(082)

УЧИТЕЉСКИ факултет на мађарском наставном језику. Међународна методичка конференција (13 ; 2024 ; Суботица)
Компетенције [Електронски извор] : зборник радова = Kompetenciák : tanulmánygyűjtemény / 13. међународна методичка конференција, Суботица, 7-8. новембар 2024. = 13. Nemzetközi Módszertani Konferencia, Szabadka, 2024. november 7-8. ; [уредник Márta Törteli Telek]. - Суботица : Учитељски факултет на мађарском наставном језику, 2024

Начин приступа (URL): https://magister.uns.ac.rs/publ/2024/978-86-81960-32-5. - Начин приступа (URL): https://magister.uns.ac.rs/Kiadvanyaink/. - Начин приступа (URL): https://magister.uns.ac.rs/Публикације/. - Насл. са насловног екрана. - Опис заснован на стању на дан 30.01.2025. - Радови на мађ., хрв. и енгл. језику. - Библиографија уз сваки рад. - Summaries.

ISBN 978-86-81960-32-5

а) Учитељи -- Образовање -- Зборници б) Васпитачи -- Образовање -- Зборници в) Учитељи -- Компетенције -- Зборници г) Васпитачи -- Компетенције -- Зборници д) Настава -- Методика -- Зборници

COBISS.SR-ID 162035721