Tartalom

Összegzés tétele

Az összegzés tételének kódjával képesek lehetünk egy adott tömb elemeit egyesével végig vizsgálni, és azok összegét egy korábban létrehozott változóban letárolni annak értékének folyamatos növelésével.

```
function Osszegzes(vizsgalandoTomb)
{
	let osszeg=0; //osszeg változó kezdőértékét beállítom 0-ra
	for(let i=0;i<vizsgalandoTomb.length;i++)
	{
		osszeg+=vizsgalandoTomb[i];
	}
	return osszeg;
}
```

Átlagszámítás tétele

Átlagszámítás szinte teljesen megegyezik az összegzés tételével, azaz végig vizsgálunk egy tömböt, és egy változóba elkezdjük összeadni a tömb elemeinek értékét. A különbség csupán annyi, hogy az eredmény megjelenítésekor, ezt az összeget el kell osztani az elemszámmal, ami nem más, mint a tömb mérete.

```
function Atlagszamitas(vizsgalandoTomb)
{
	let osszeg=0;
	for(let i=0;i<vizsgalandoTomb.length;i++)
	{
		osszeg+=vizsgalandoTomb[i];
	}
	return osszeg/vizsgalandoTomb.length;
}
```

Megszámlálás tétele

Megszámlálás tételével, mint a legtöbb programozási tétellel, végignézzük a tömb elemeit és megszámoljuk, hogy hány darab t tulajdonságú elem van benne. Ehhez nincs másra szükségünk, mint egy számlálóra, valamint a tömb bejárására használt ciklusban egy feltételre, ami tartalmazza a t tulajdonságot. (feltétel lehet pl.: a páros számok megszámlálása)

```
function megszamlalasTeteleFuggveny(tomb)
{
	let darab=0;
	for(let i=0;i<tomb.length;i++)
	{
		if(tomb[i]%2==0)
		{
			darab++; //darab=darab+1;
		}
	}
  return darab;
}
```

Kiválogatás tétele

Kiválogatás tétele esetén hasonlóan járunk el, mint megszámlálás tételekor, keressük az adott tömbben a t tulajdonsággal rendelkező elemeket, ám ahelyett, hogy ezeket egyszerűen csak megszámolnánk, kiírhatjuk a képernyőre, esetleg kimenthetjük egy másik összetett változóba azt. Ilyenkor csupán egy felteltére van szükségünk a vizsgálatot végző ciklusban. (feltétel lehet pl.: a páros számok megszámlálása)

```
function kivalogatasTeteleFuggveny(tomb)
{
	 let kivalasztottElemek=[];
	for(let i=0;i<tomb.length;i++)
	{
		if(tomb[i]%2==0)
		{
			kivalasztottElemek.push(tomb[i]);
		}
	}
	return kivalasztottElemek;
}
```

Minimum kiválasztás tétele

Minimum kiválasztás esetén keressük a tömbben megtalálható legkisebb elemet, arra viszont ügyelnünk kell, hogy a keresést segítő minimum értéket tároló változó értéke, sosem lehet kisebb, mint a tömb legkisebb eleme, mert így az sosem fog módosulni. Érdemes a használatához a tömb egyik elemének értékét kezdőértéknek meghatározni, majd ezután megvizsgálni a tömb többi elemét, és ha találunk az adott minimum értéknél kisebbet, módosítjuk azt az éppen vizsgált értékre. Esetleg tárolhatjuk a legkisebb értékű változónak a helyét is, az alábbi módon.

Érték alapján

```
function MinErtekKeresesFuggveny(tomb)
{
	let minErtek=tomb[0];
	for(let i=1;i<tomb.length;i++)
	{
		if(tomb[i]<minErtek)
		{
			minErtek=tomb[i];
		}
	}
	return minErtek;
}
```

Index alapján

```
function MinIndexKeresesFuggveny(tomb)
{
	let minIndex=0;
	for(let i=1;i<tomb.length;i++)
	{
		if(tomb[i]<tomb[minIndex])
		{
			minIndex=i;
		}
	}
	return minIndex;
}
```

Maximum kiválasztás tétele

Maximum kiválasztás esetén keressük a tömbben megtalálható legnagyobb elemet, arra viszont ügyelnünk kell, hogy a keresést segítő maximum értéket tároló változó értéke, sosem lehet nagyobb, mint a tömb legnagyobb eleme, mert így az sosem fog módosulni. Érdemes a használatához a tömb egyik elemének értékét kezdőértéknek meghatározni, majd ezután megvizsgálni a tömb többi elemét, és ha találunk az adott maximum értéknél nagyobbat, módosítjuk azt az éppen vizsgált értékre. Esetleg tárolhatjuk a legnagyobb értékű változónak a helyét is, az alábbi módon.

Érték alapján

```
function MaxErtekKeresesFuggveny(tomb)
{
	let maxErtek=tomb[0];
	for(let i=1;i<tomb.length;i++)
	{
		if(tomb[i]>maxErtek)
		{
			maxErtek=tomb[i];
		}
	 }
	return maxErtek;
}
```

Index alapján

```
function MaxIndexKeresesFuggveny(tomb)
{
	let maxIndex=0;
	for(let i=1;i<tomb.length;i++)
	{
		if(tomb[i]>tomb[maxIndex])
		{
			maxIndex=i;
		}
	}
	return maxIndex;
}
```

Halmazműveletek

Említést kell tenni még a halmazműveletekről is, itt egy összetett adatszerkezet helyett többet hasonlítunk össze. (Itt érdemes tömbök helyett listákat használni, mivel nem tudhatjuk, hány elem fog majd a végén belekerülni)

UNIÓ esetén két adott lista **(a, b)** elemeit mentjük egy harmadikba **(unio),** ügyelve arra, hogy amennyiben a listában vizsgált elem már szerepelt korábban az unió listában, azt ismételten ne írjuk bele.

METSZET létrehozása alkalmával, megvizsgáljuk, hogy az első **(a)** lista elemeit egyesével, hogy szerepelnek-e a második **(b)** listában. Amennyiben egyezést találunk, beleilleszthetjük az a metszet listába, hogyha még nem szerepelt korábban.

KÜLÖNBSÉG alkalmával pedig, hasonlóan járunk el mint a metszetnél, de míg ott a közös elemeket keressük, itt egy elem akkor kerül csak a különbség listába, ha az a másikban nem található meg.

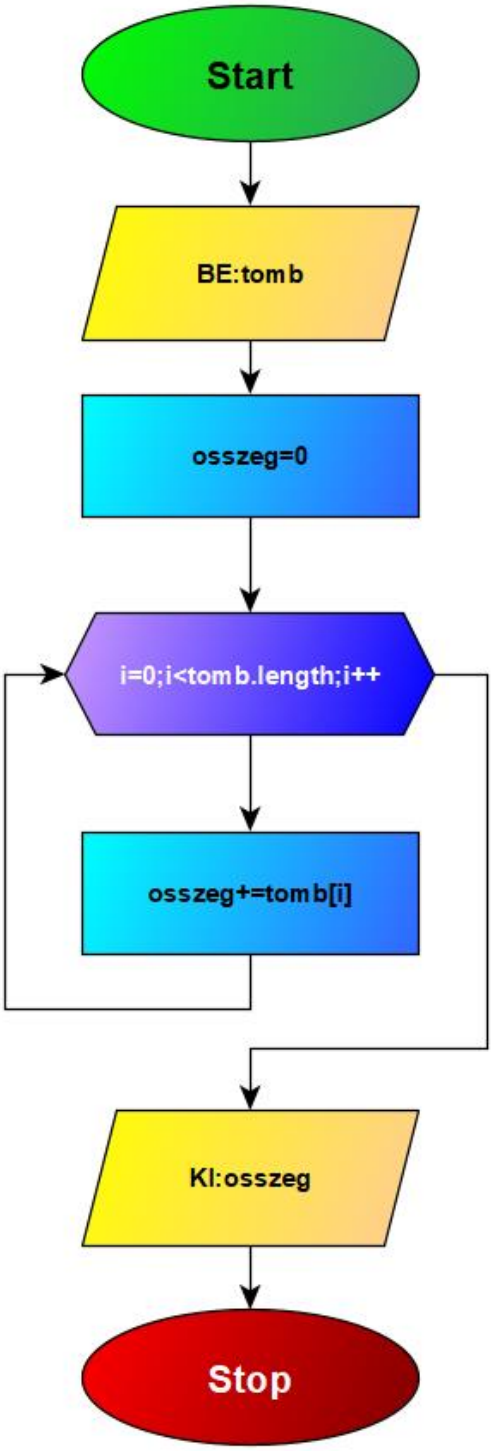
UNIO

```
function Unio(tomb1, tomb2){
	let unio=[];
	//Első halmaz elemein megy végig
	for(let i=0;i<tomb1.length;i++)
	{
		let szerepelE=false;
		 for(let j=0;j<unio.length;j++)
		{
			if(tomb1[i]==unio[j])
			{
				szerepelE=true;
			}
		 }
		 if(szerepelE==false){
			unio.push(tomb1[i]);
		}
	}
	//Második halmaz elemein megy végig
	for(let i=0;i<tomb2.length;i++)
	{
		let szerepelE=false;
		for(let j=0;j<unio.length;j++)
		{
			if(tomb2[i]==unio[j])
			{
				szerepelE=true;
			}
		 }
		 if(szerepelE==false){
			unio.push(tomb2[i]);
		}
	}
	return unio;
}
```

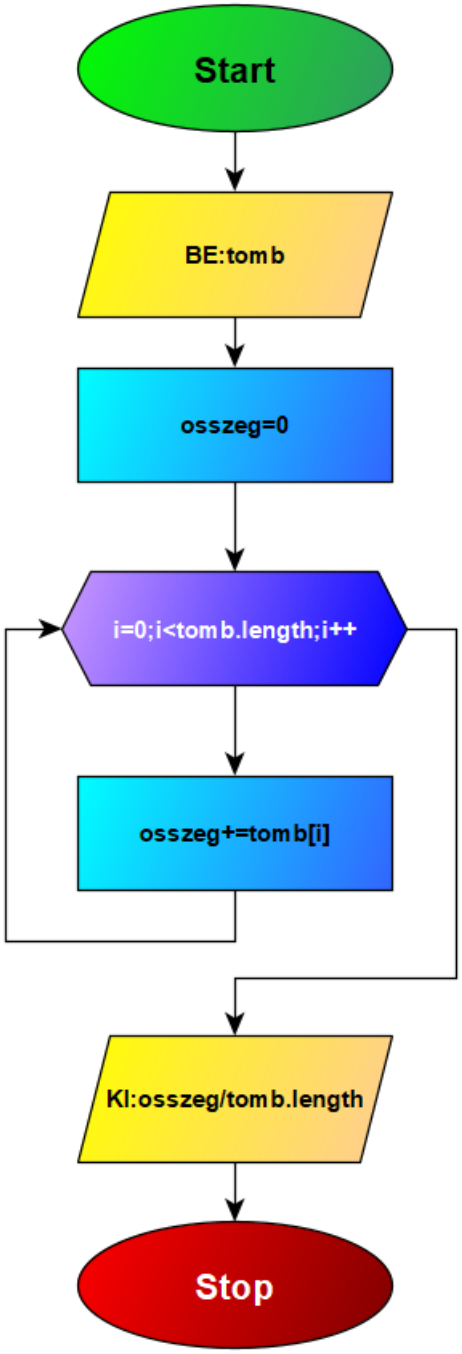
METSZET

```
function Metszet(tomb1, tomb2){
	let metszet= [];
	for(let i=0;i<tomb1.length;i++)
	{
		for(let j=0;j<tomb2.length;j++)
		{
			if(tomb1[i]==tomb2[j])
			{
				let szerepelE=false;
				for(let k=0;k<metszet.length;k++)
				{
					if(tomb1[i]==metszet[k])
					{
						szerepelE=true;
					}
				 }
				 if(szerepelE==false)
				{
					metszet.push(tomb1[i]);
				}
			 }
		}
	}
	return metszet;
}
```

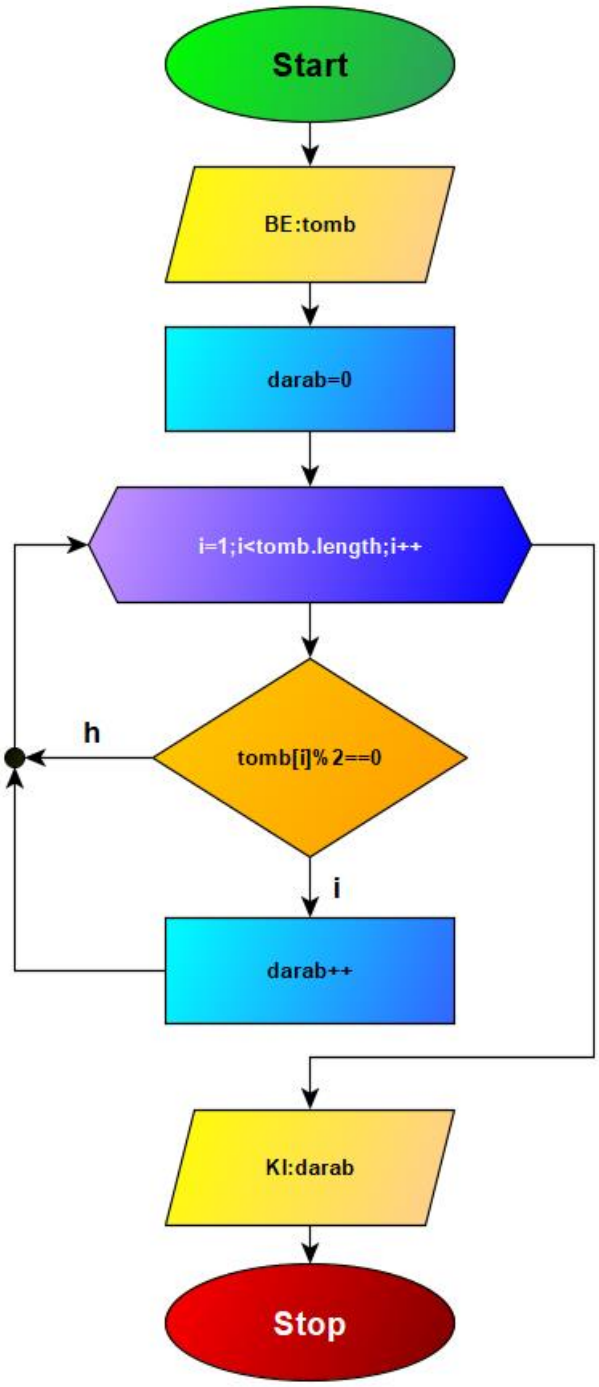
Ábrajegyzék, folyamatábrák



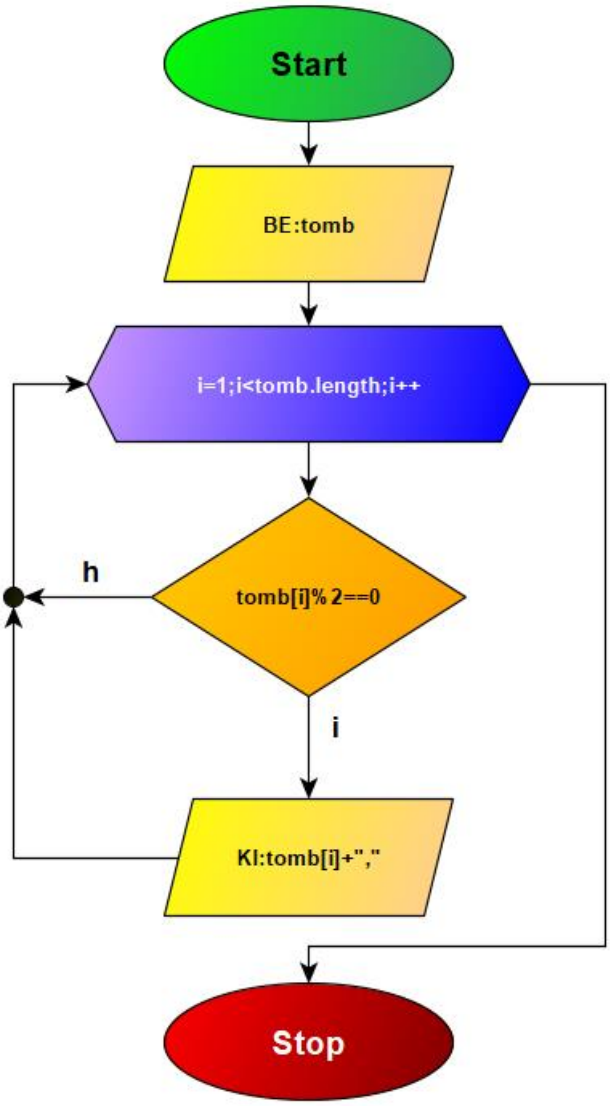
1. ábra Összegzés tétele folyamatábra



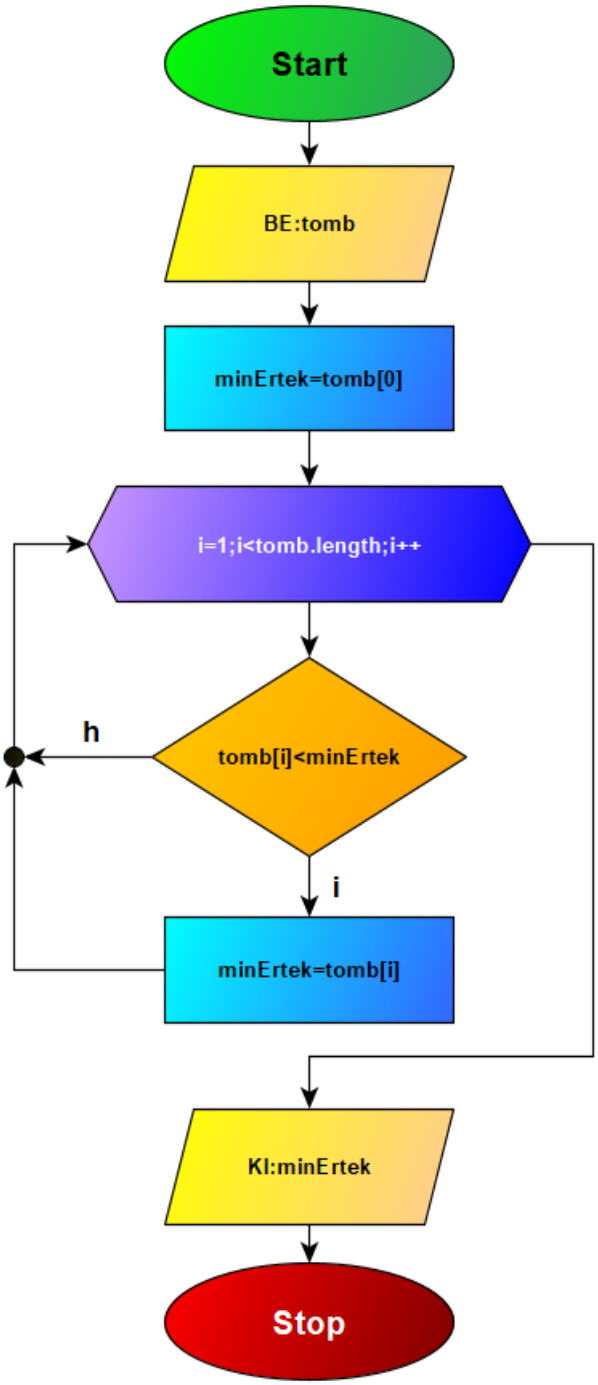
2. ábra Átlagszámítás tétele



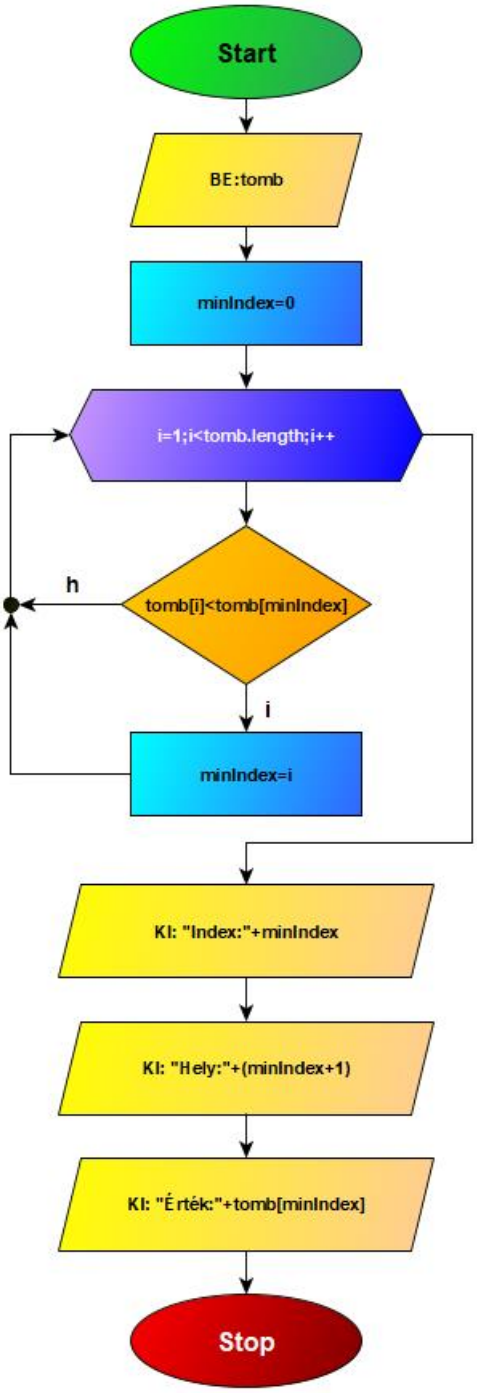
3. ábra Megszámlálás tétele



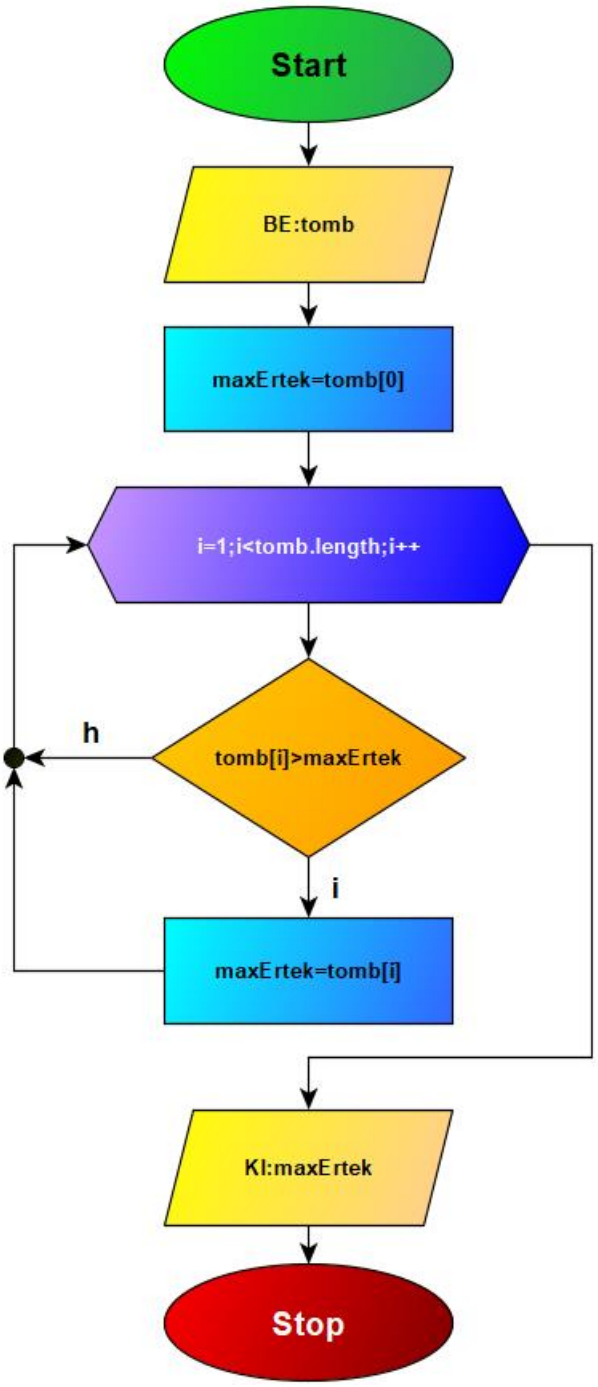
4. ábra Kiválogatás tétele



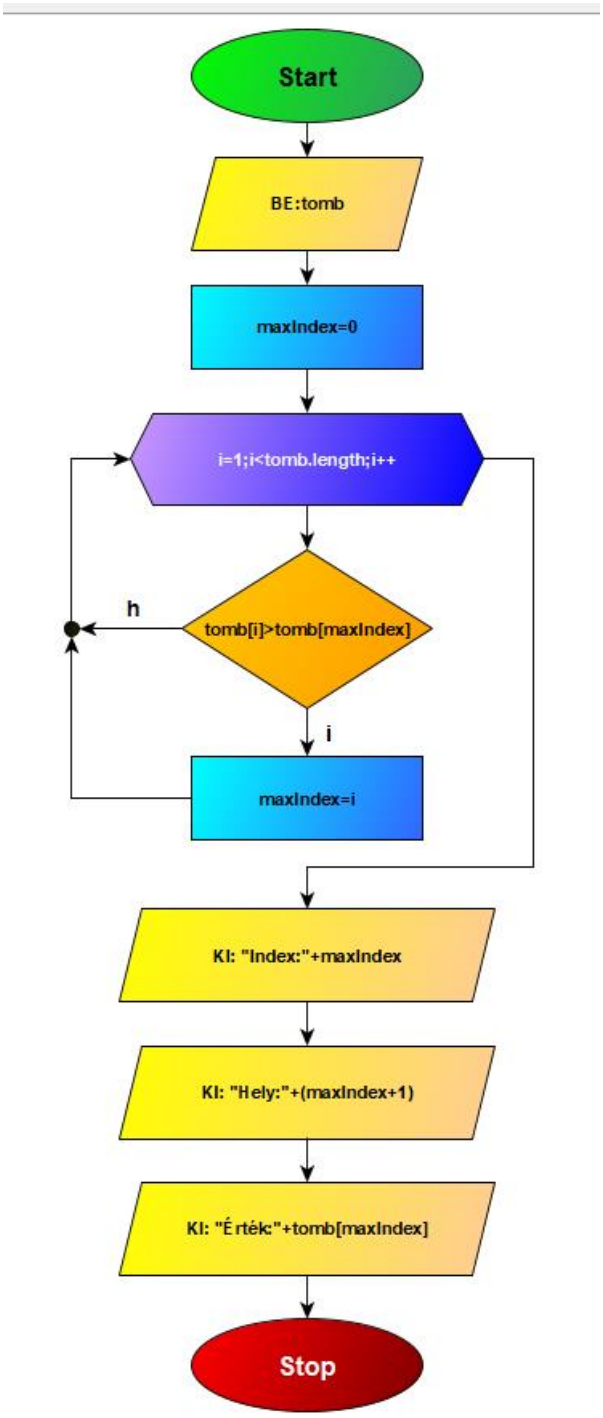
5. ábra Minimum keresés tétele érték alapján



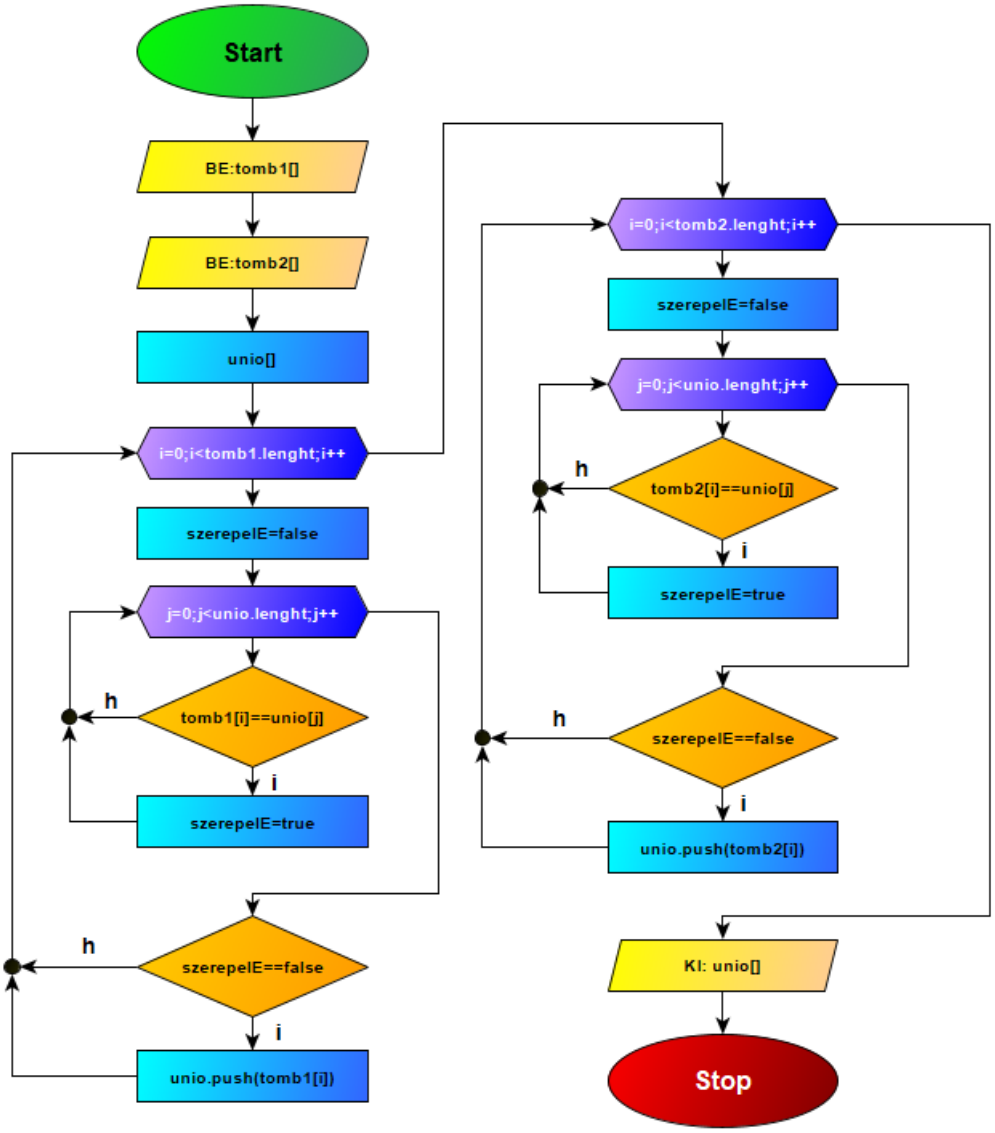
6. ábra Minimum keresés tétele index alapján



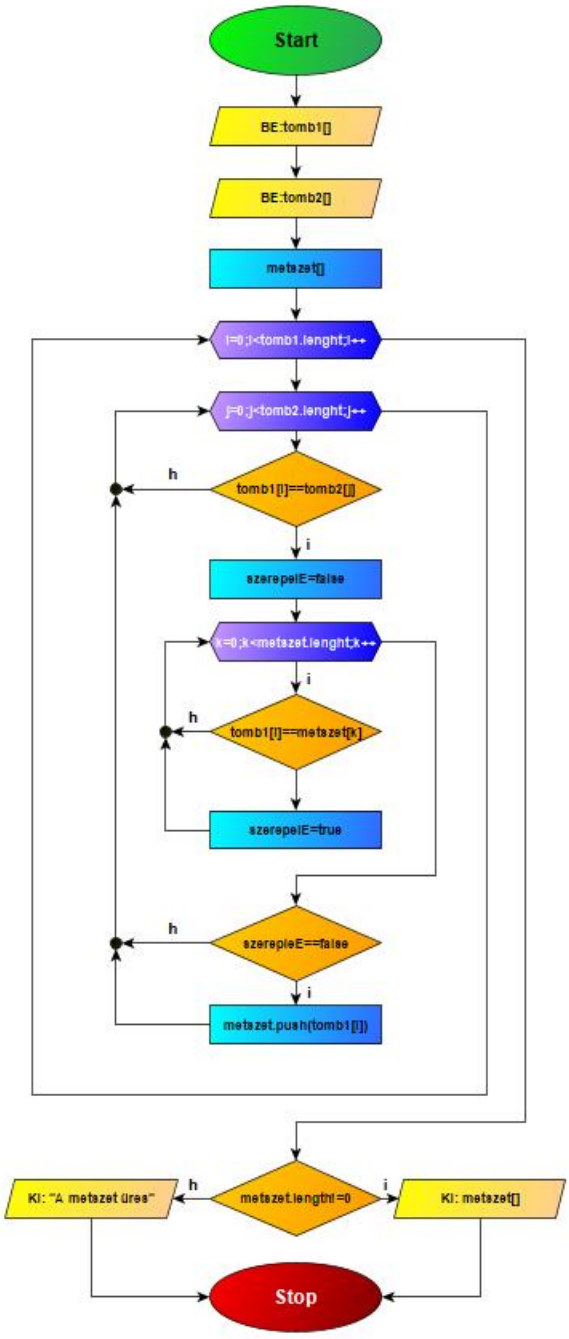
7. ábra Maximum keresés tétele érték alapján



8. ábra Maximum keresés tétele index alapján



9. ábra Halmazművelet: UNIÓ



10. ábra Halmazművelet: METSZET