**### 7. óra: Vezérlési szerkezetek II.**

**- Ciklusok: `for`, `while`, `do while`**

**- Gyakorlati feladatok: egyszerű ciklusok készítése és gyakorlása.**

**7. óra: Vezérlési szerkezetek II. (Ciklusok: for, while, do while)**

**Óravázlat – 45 perc**

**Bevezetés (5 perc)**

* *Tanár:* „Üdvözlök mindenkit! A mai órán a vezérlési szerkezetek másik fontos csoportjával, a ciklusokkal fogunk foglalkozni. Ezek a szerkezetek lehetővé teszik, hogy egy adott műveletet többször végrehajtsunk, például egy lista elemein végighaladva, vagy egy feltétel teljesüléséig ismételjünk egy folyamatot.”
* *Tanár:* „Képzeljétek el, hogy szeretnétek egy játékban egy karaktert mozgatni, vagy többször szeretnétek megjeleníteni egy üzenetet a képernyőn. A ciklusokkal mindezt egyszerűen megtehetjük!”

**for ciklus bemutatása (10 perc)**

1. *Tanár:* „Kezdjük a for ciklussal, amely az egyik leggyakrabban használt ciklusszerkezet. A for ciklus egy adott számú ismétlést tesz lehetővé. Nézzünk meg egy egyszerű példát, amely a számokat 1-től 5-ig kiírja.”
   * *Mutasd be a kódot:*

javascript

Copy code

for (let i = 1; i <= 5; i++) {

console.log(i);

}

1. *Tanár:* „Ez a for ciklus három részből áll: a kezdeti értékadás (let i = 1), a feltétel (i <= 5), és a léptetés (i++). Lépésről lépésre nézzük meg, hogyan működik ez a ciklus!”
   * *Részletes magyarázat:*
     + „Az i változó értéke kezdetben 1. Ellenőrizzük a feltételt: i <= 5. Ez igaz, tehát a ciklus belső blokkja végrehajtódik, és a konzolon megjelenik az 1.”
     + „Ezután a léptetés következik: i++, ami 1-gyel növeli az i értékét. Így az i most 2 lesz. A ciklus újra ellenőrzi a feltételt, és mivel 2 <= 5 igaz, ismét lefut a blokk.”
     + „Ez a folyamat addig ismétlődik, amíg a feltétel hamis nem lesz, vagyis az i értéke meghaladja az 5-öt.”
2. *Tanár:* „Kérdés: Mi történik, ha a <= helyett <-et használunk a feltételben? Próbáljuk ki!”
   * *Kód módosítása és megbeszélése.*

**Gyakorlati Feladat (5 perc)**

* *Tanár:* „Írjatok egy for ciklust, amely a 10 és 1 közötti számokat írja ki visszafelé!”
  + Segítség az osztály munkájában, kérdések megválaszolása.

**while ciklus bemutatása (10 perc)**

1. *Tanár:* „Most nézzük meg a while ciklust, amely addig fut, amíg egy adott feltétel igaz. Nézzünk egy példát, amely 1-től 5-ig számol.”
   * A képen képernyőkép, szöveg látható

     Automatikusan generált leírás*Példa:*

javascript

Copy code

let i = 1;

while (i <= 5) {

console.log(i);

i++;

}

1. *Tanár:* „A while ciklus a feltétel (i <= 5) igazságát vizsgálja minden iteráció elején. Ha igaz, a ciklus belső blokkja végrehajtódik, és a console.log() kiírja az i értékét. Az i++ léptetés itt a ciklus végén történik meg. Ha a feltétel hamis, a ciklus leáll.”
2. *Részletes magyarázat:*
   * „Ez a szerkezet akkor nagyon hasznos, ha nem tudjuk előre, hogy hány alkalommal fog lefutni a ciklus, például egy felhasználói bemenet feldolgozásánál.”

**Gyakorlati Feladat (5 perc)**

* *Tanár:* „Készítsetek egy while ciklust, amely addig fut, amíg egy változó értéke kisebb, mint 20, és minden iterációban növelje az értéket 3-mal!”
  + Segítség az osztályban.

**do while ciklus bemutatása (10 perc)**

1. *Tanár:* „A do while ciklus hasonlít a while ciklushoz, de van egy lényeges különbség: a belső blokk mindig legalább egyszer lefut, mert először a műveletet hajtja végre, és csak utána ellenőrzi a feltételt.”
   * *Példa:*

*A képen képernyőkép, szöveg, Multimédiás szoftver, szoftver látható

Automatikusan generált leírás*

javascript

Copy code

let i = 1;

do {

console.log(i);

i++;

} while (i <= 5);

1. *Tanár:* „Lépésről lépésre haladva: először a blokk fut le, kiírja az i értékét, majd növeli azt 1-gyel. Ezután ellenőrzi a feltételt. Ha igaz, újra végrehajtódik; ha hamis, a ciklus befejeződik.”
2. *Tanár:* „Fontos különbség, hogy akkor is lefut legalább egyszer a blokk, ha a feltétel már az elején hamis. Próbáljuk ki egy példán keresztül, amikor az i kezdeti értéke 10!”
   * *Kód módosítása és megbeszélése.*

**Gyakorlati Feladat (5 perc)**

* *Tanár:* „Írjatok egy do while ciklust, amely egyszer végrehajtja a blokkot, függetlenül attól, hogy a feltétel igaz-e. Például próbáljuk ki azt, hogy egy értéket összehasonlítunk egy kicsi számmal, de a blokk mindig végrehajtódik legalább egyszer.”

**Óraösszegzés (5 perc)**

* *Tanár:* „Ma megismertük a for, while és do while ciklusokat. Ezekkel az ismétlődő folyamatokat kezelhetjük, például listák elemein végezhetünk műveleteket vagy feltételek teljesüléséig ismételhetünk egy folyamatot.”
* *Tanár:* „Van-e bármilyen kérdésetek? Ha bármi nem világos, bátran kérdezzetek!”

### 7. óra: Vezérlési szerkezetek II. (Ciklusok: `for`, `while`, `do while`)

---

#### \*\*Óravázlat – 45 perc\*\*

\*\*Bevezetés (5 perc)\*\*

- \*Tanár:\* „Üdvözlök mindenkit! A mai órán a vezérlési szerkezetek másik fontos csoportjával, a ciklusokkal fogunk foglalkozni. Ezek a szerkezetek lehetővé teszik, hogy egy adott műveletet többször végrehajtsunk, például egy lista elemein végighaladva, vagy egy feltétel teljesüléséig ismételjünk egy folyamatot.”

- \*Tanár:\* „Képzeljétek el, hogy szeretnétek egy játékban egy karaktert mozgatni, vagy többször szeretnétek megjeleníteni egy üzenetet a képernyőn. A ciklusokkal mindezt egyszerűen megtehetjük!”

---

\*\*`for` ciklus bemutatása (10 perc)\*\*

1. \*Tanár:\* „Kezdjük a `for` ciklussal, amely az egyik leggyakrabban használt ciklusszerkezet. A `for` ciklus egy adott számú ismétlést tesz lehetővé. Nézzünk meg egy egyszerű példát, amely a számokat 1-től 5-ig kiírja.”

- \*Mutasd be a kódot:\*

```javascript

for (let i = 1; i <= 5; i++) {

console.log(i);

}

```

2. \*Tanár:\* „Ez a `for` ciklus három részből áll: a kezdeti értékadás (`let i = 1`), a feltétel (`i <= 5`), és a léptetés (`i++`). Lépésről lépésre nézzük meg, hogyan működik ez a ciklus!”

- \*Részletes magyarázat:\*

- „Az `i` változó értéke kezdetben 1. Ellenőrizzük a feltételt: `i <= 5`. Ez igaz, tehát a ciklus belső blokkja végrehajtódik, és a konzolon megjelenik az 1.”

- „Ezután a léptetés következik: `i++`, ami 1-gyel növeli az `i` értékét. Így az `i` most 2 lesz. A ciklus újra ellenőrzi a feltételt, és mivel `2 <= 5` igaz, ismét lefut a blokk.”

- „Ez a folyamat addig ismétlődik, amíg a feltétel hamis nem lesz, vagyis az `i` értéke meghaladja az 5-öt.”

3. \*Tanár:\* „Kérdés: Mi történik, ha a `<=` helyett `<`-et használunk a feltételben? Próbáljuk ki!”

- \*Kód módosítása és megbeszélése.\*

\*\*Gyakorlati Feladat (5 perc)\*\*

- \*Tanár:\* „Írjatok egy `for` ciklust, amely a 10 és 1 közötti számokat írja ki visszafelé!”

- Segítség az osztály munkájában, kérdések megválaszolása.

---

\*\*`while` ciklus bemutatása (10 perc)\*\*

1. \*Tanár:\* „Most nézzük meg a `while` ciklust, amely addig fut, amíg egy adott feltétel igaz. Nézzünk egy példát, amely 1-től 5-ig számol.”

- \*Példa:\*

```javascript

let i = 1;

while (i <= 5) {

console.log(i);

i++;

}

```

2. \*Tanár:\* „A `while` ciklus a feltétel (`i <= 5`) igazságát vizsgálja minden iteráció elején. Ha igaz, a ciklus belső blokkja végrehajtódik, és a `console.log()` kiírja az `i` értékét. Az `i++` léptetés itt a ciklus végén történik meg. Ha a feltétel hamis, a ciklus leáll.”

3. \*Részletes magyarázat:\*

- „Ez a szerkezet akkor nagyon hasznos, ha nem tudjuk előre, hogy hány alkalommal fog lefutni a ciklus, például egy felhasználói bemenet feldolgozásánál.”

\*\*Gyakorlati Feladat (5 perc)\*\*

- \*Tanár:\* „Készítsetek egy `while` ciklust, amely addig fut, amíg egy változó értéke kisebb, mint 20, és minden iterációban növelje az értéket 3-mal!”

- Segítség az osztályban.

---

\*\*`do while` ciklus bemutatása (10 perc)\*\*

1. \*Tanár:\* „A `do while` ciklus hasonlít a `while` ciklushoz, de van egy lényeges különbség: a belső blokk mindig legalább egyszer lefut, mert először a műveletet hajtja végre, és csak utána ellenőrzi a feltételt.”

- \*Példa:\*

```javascript

let i = 1;

do {

console.log(i);

i++;

} while (i <= 5);

```

2. \*Tanár:\* „Lépésről lépésre haladva: először a blokk fut le, kiírja az `i` értékét, majd növeli azt 1-gyel. Ezután ellenőrzi a feltételt. Ha igaz, újra végrehajtódik; ha hamis, a ciklus befejeződik.”

3. \*Tanár:\* „Fontos különbség, hogy akkor is lefut legalább egyszer a blokk, ha a feltétel már az elején hamis. Próbáljuk ki egy példán keresztül, amikor az `i` kezdeti értéke 10!”

- \*Kód módosítása és megbeszélése.\*

\*\*Gyakorlati Feladat (5 perc)\*\*

- \*Tanár:\* „Írjatok egy `do while` ciklust, amely egyszer végrehajtja a blokkot, függetlenül attól, hogy a feltétel igaz-e. Például próbáljuk ki azt, hogy egy értéket összehasonlítunk egy kicsi számmal, de a blokk mindig végrehajtódik legalább egyszer.”

---

\*\*Óraösszegzés (5 perc)\*\*

- \*Tanár:\* „Ma megismertük a `for`, `while` és `do while` ciklusokat. Ezekkel az ismétlődő folyamatokat kezelhetjük, például listák elemein végezhetünk műveleteket vagy feltételek teljesüléséig ismételhetünk egy folyamatot.”

- \*Tanár:\* „Van-e bármilyen kérdésetek? Ha bármi nem világos, bátran kérdezzetek!”

Feladatok:  
  
Természetesen! Íme néhány gyakorlati feladat a hetedik órára, amelyek a ciklusok gyakorlását célozzák meg különböző nehézségi szinteken:

---

### \*\*Gyakorlati Feladatok – `for` ciklus\*\*

1. \*\*Egyszerű számlálás\*\*

- Írj egy `for` ciklust, amely kiírja a számokat 1-től 10-ig a konzolra!

2. \*\*Páros számok listázása\*\*

- Készíts egy `for` ciklust, amely az 1 és 20 közötti összes páros számot kiírja!

3. \*\*Szorzótábla\*\*

- Írj egy `for` ciklust, amely a 7-es szorzótábla első 10 elemét jeleníti meg (pl. 1 \* 7, 2 \* 7, stb.)!

4. \*\*Összegszámítás\*\*

- Készíts egy `for` ciklust, amely összeadja az 1-től 100-ig terjedő számokat, és kiírja az eredményt!

---

### \*\*Gyakorlati Feladatok – `while` ciklus\*\*

1. \*\*Egyszerű számlálás `while` ciklussal\*\*

- Írj egy `while` ciklust, amely kiírja az 1-től 10-ig terjedő számokat!

2. \*\*Felhasználói bemenet kezelése\*\*

- Készíts egy `while` ciklust, amely addig kér be számokat a felhasználótól, amíg az nem ad meg egy 0 értéket! A végén írd ki az összesített összeget.

3. \*\*Tippelős játék\*\*

- Hozz létre egy `while` ciklust, amely egy előre meghatározott számot próbáltat kitalálni a felhasználóval! A ciklus addig fusson, amíg a felhasználó el nem találja a számot. Adj visszajelzést, hogy a tipp nagyobb vagy kisebb a keresett számnál.

---

### \*\*Gyakorlati Feladatok – `do while` ciklus\*\*

1. \*\*Legalább egyszeri futás\*\*

- Készíts egy `do while` ciklust, amely egy változó értékét 1-gyel növeli, és a végén kiírja az értéket. A ciklus akkor álljon le, ha a változó értéke eléri a 10-et.

2. \*\*Szorzótábla ismétlés\*\*

- Írj egy `do while` ciklust, amely először kiír egy 5-ös szorzótáblát, majd rákérdez, hogy szeretnéd-e megismételni (igen/nem). Ha a válasz "igen", a ciklus újra végrehajtódik.

---

### \*\*Kombinált Feladatok\*\*

1. \*\*Fibonacci-sorozat generálása\*\*

- Írj egy `for` vagy `while` ciklust, amely kiszámolja a Fibonacci-sorozat első 10 elemét, és kiírja őket a konzolra!

2. \*\*Tömb elemeinek összege\*\*

- Hozz létre egy tömböt tetszőleges számokkal, és használj egy `for` ciklust az elemek összegzésére. A végén írd ki az összeget.

3. \*\*Prime számok keresése 1-től 100-ig\*\*

- Írj egy `for` ciklust, amely az 1 és 100 közötti összes prímszámot megtalálja és kiírja. Használj beágyazott ciklust a vizsgálathoz.

4. \*\*Szöveg manipulálása ciklusokkal\*\*

- Adj meg egy sztringet, és írj egy `for` ciklust, amely visszafelé kiírja a sztring karaktereit.

5. \*\*Számok összeadása felhasználói bevitel alapján\*\*

- Kérd meg a felhasználót, hogy adjon meg számokat egy `while` ciklus segítségével. Az adatok bevitele addig folytatódjon, amíg a felhasználó nem ír be egy negatív számot. A ciklus után írd ki a beírt számok összegét.

---

Ezek a feladatok változatos módon segítik a `for`, `while` és `do while` ciklusok mélyebb megértését, miközben lehetőséget biztosítanak az algoritmikus gondolkodás fejlesztésére is.

Kidolgozások

Persze, megoldom a feladatokat! Itt vannak a megoldások mindegyik gyakorlati feladatra:

---

### \*\*Megoldások – `for` ciklusos feladatok\*\*

1. \*\*Egyszerű számlálás\*\*



```javascript

for (let i = 1; i <= 10; i++) {

console.log(i);

}

```

1. \*\*Páros számok listázása\*\*

A képen szöveg, képernyőkép, Betűtípus, fekete látható

Automatikusan generált leírás

```javascript

for (let i = 2; i <= 20; i += 2) {

console.log(i);

}

```

1. \*\*Szorzótábla\*\*

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

```javascript

for (let i = 1; i <= 10; i++) {

console.log(`${i} \* 7 = ${i \* 7}`);

}

```

1. \*\*Összegszámítás\*\*

A képen képernyőkép, szöveg látható

Automatikusan generált leírás

```javascript

let sum = 0;

for (let i = 1; i <= 100; i++) {

sum += i;

}

console.log("Az összeg: " + sum);

```

---

### \*\*Megoldások – `while` ciklusos feladatok\*\*

1. \*\*Egyszerű számlálás `while` ciklussal\*\*

A képen szöveg, képernyőkép, Betűtípus, Multimédiás szoftver látható

Automatikusan generált leírás

```javascript

let i = 1;

while (i <= 10) {

console.log(i);

i++;

}

```

1. \*\*Felhasználói bemenet kezelése\*\*

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

```javascript

let sum = 0;

let input;

while (input !== 0) {

input = parseInt(prompt("Adj meg egy számot (0 a kilépéshez): "), 10);

if (input !== 0) {

sum += input;

}

}

console.log("Az összeg: " + sum);

```

1. \*\*Tippelős játék\*\*

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

```javascript

const targetNumber = 15; // Ez a titkos szám, amit ki kell találni

let guess;

while (guess !== targetNumber) {

guess = parseInt(prompt("Tippelj egy számot: "), 10);

if (guess < targetNumber) {

console.log("Túl alacsony!");

} else if (guess > targetNumber) {

console.log("Túl magas!");

} else {

console.log("Gratulálok, eltaláltad!");

}

}

```

---

### \*\*Megoldások – `do while` ciklusos feladatok\*\*

1. \*\*Legalább egyszeri futás\*\*

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

```javascript

let i = 1;

do {

console.log(i);

i++;

} while (i <= 10);

```

1. \*\*Szorzótábla ismétlés\*\*

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

```javascript

let repeat;

do {

for (let i = 1; i <= 10; i++) {

console.log(`5 \* ${i} = ${5 \* i}`);

}

repeat = prompt("Szeretnéd újra megismételni? (igen/nem)").toLowerCase();

} while (repeat === "igen");

```

---

### \*\*Megoldások – Kombinált feladatok\*\*

1. \*\*Fibonacci-sorozat generálása\*\*

A képen szöveg, képernyőkép, Betűtípus, szoftver látható

Automatikusan generált leírás

```javascript

let a = 0, b = 1;

console.log(a);

console.log(b);

for (let i = 2; i < 10; i++) {

let c = a + b;

console.log(c);

a = b;

b = c;

}

```

1. \*\*Tömb elemeinek összege\*\*

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

```javascript

const numbers = [1, 2, 3, 4, 5];

let sum = 0;

for (let i = 0; i < numbers.length; i++) {

sum += numbers[i];

}

console.log("Az összeg: " + sum);

```

1. \*\*Prime számok keresése 1-től 100-ig\*\*

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

```javascript

for (let i = 2; i <= 100; i++) {

let isPrime = true;

for (let j = 2; j <= Math.sqrt(i); j++) {

if (i % j === 0) {

isPrime = false;

break;

}

}

if (isPrime) {

console.log(i);

}

}

```

1. \*\*Szöveg manipulálása ciklusokkal\*\*

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

```javascript

const str = "Hello, világ!";

let reversed = "";

for (let i = str.length - 1; i >= 0; i--) {

reversed += str[i];

}

console.log(reversed);

```

1. \*\*Számok összeadása felhasználói bevitel alapján\*\*

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

```javascript

let sum = 0;

let number;

do {

number = parseInt(prompt("Adj meg egy számot (negatív szám a kilépéshez): "), 10);

if (number >= 0) {

sum += number;

}

} while (number >= 0);

console.log("Az összeg: " + sum);

```

---

Ezek a megoldások különböző típusú ciklusok és feltételek használatával segítik a JavaScript vezérlési szerkezeteinek mélyebb megértését. Ha van további kérdésed vagy módosítási igényed, jelezd bátran!