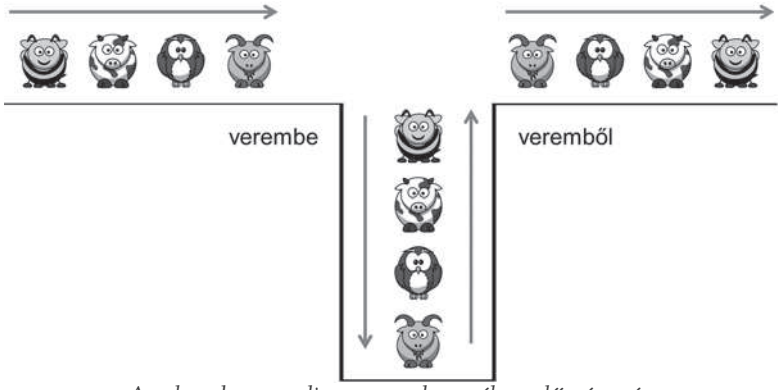
17. A verem

A verem adattípus

Ha hátizsákba pakoljuk egy túra kellékeit, akkor ügyelünk arra, hogy a leggyakrabban szükséges eszközök felülre kerüljenek. A zsák alján elhelyezkedő felszerelésünkhöz csak akkor férünk hozzá, ha kivesszük a fölöttük lévő holmit. Közel hasonló sorrendet követ egy liftbe történő beszállás, illetve kiszállás rendje. Akkor tudunk könnyen kiszállni először, ha utoljára szálltunk be.

Az informatikában is gyakran találkozunk ilyen elrendezéssel. Ha eljárások hívják egymást, akkor az utoljára induló eljárás fejeződik be először. Az előző eljárás folytatásához szükséges visszatérési cím egy verembe kerül, ahonnan könnyen elő lehet venni. Említhetnénk továbbá a szabályos zárójelezést (az utoljára megnyitott zárójelet zárjuk be először), az egymásba ágyazott ciklusok befejeződésének sorrendjét stb.

Verem (stack): olyan lineáris kollekció, melynél az elemeket a betétellel ellentétes sorrendben tudjuk kivenni.



Az elemek sorrendje a verem használata előtt és után

A vermet gyakran LIFO adatszerkezetnek nevezzük (Last In First Out: utoljára be, először ki). Az angol szakkifejezések szerint a *push* művelet a verembe helyezést, a *pop* pedig a veremből való kivételt jelenti.

A modern programozási nyelvek ismerik a verem adatszerkezetet. A leggyakrabban objektummal valósítják meg. Ennek hiányában például egydimenziós tömbbel szimulálhatjuk egy verem működését. A verembe helyezett elem az eddigi elemek mögé kerül a tömbben, és onnan is vesszük (olvassuk) ki. Közben nyilván kell tartanunk a verem „tetejét". Ez lehet az utoljára betett elem vagy pedig az első üres hely indexe.

A vermet láncolt adatszerkezettel szintén megvalósíthatjuk. Ez utóbbit tankönyvünkben később ismertetjük.

A verem adatszerkezetet más adattípusok (például gráfok) feldolgozásánál is alkalmazzuk.


Programozási összefoglaló: A verem adatszerkezet

Verem létrehozása

Ha a verem objektum, akkor deklarálunk egy rá hivatkozó változót, majd az *Új* operátorral létrehozzuk az objektumot. Mindkét utasításban megadjuk a verem elemeinek típusát:

Változó *Veremnév* Mint Verem(Típus: *Elemtípus*)
Veremnév = Új Verem(Típus: *Elemtípus*)

Az elemek típusa általában tetszőleges, előzőleg már definiált adattípus lehet. A létrehozás után a verem üres, tehát nem tartalmaz elemet.

A verem létrehozásakor megadhatjuk az elemeit. A következő két utasítás létrehoz egy vermet, majd átmásolja a verembe a zárójelben megadott, felsorolható típusú kollekció elemeit:

```
Dim Változónév As Stack(Of Elemtípus)
Változónév = New Stack(Of Elemtípus)(felsorolható kollekció)
```

Az inicializálásnál ügyeljünk az elemek sorrendjére!

1. gyakorlat. Készítsünk programot, amelyben deklarálunk és létrehozunk egy egész számok tárolására alkalmas vermet! Töltsük fel néhány adattal!

A verem tulajdonságai és metódusai

A *Verembe* metódussal új elemet helyezünk a verem tetejére:

Változónév.Verembe(*kifejezés*)

A *Veremből* metódussal elemet veszünk ki a veremből. A metódus visszatérési értéke a verem tetején lévő elem, amit egyben töröl is a veremből:

Változónév.Veremből()

Ha a kivett elemet nem szerepeltetjük egy értékadó utasításban, akkor elveszítjük. A metódusok használatakor tartsuk szem előtt, hogy az utoljára betett (azaz a verem tetején lévő) elemet vesszük ki először!

Elem kivétele előtt meg kell győződnünk arról, hogy a verem nem üres. Erre az *Üres-e* metódus ad lehetőséget, melynek visszatérési értéke üres verem esetén *Igaz*, egyébként pedig *Hamis*:

Változónév.Üres-e()

Az *Üres-e* metódus helyett egyes programozási nyelvek lehetővé teszik az elemszám tulajdonság lekérdezését:

Változónév.Elemszám

Korlátozott méret (például tömbben tárolt verem) esetén szükség lehet egy *Tele-e* metódusra. A verembe csak akkor tehetünk új elemet, ha a metódus visszatérési értéke *Hamis*.

A verem tetején lévő elemet a *Tető* metódus visszatérési értéke adja meg. A metódus nem módosítja a vermet, tehát benne hagyja az elemet:

Változónév.Tető()

Bár nem szabványos veremművelet, a programozási nyelvek mégis gyakran megengedik, hogy elérjük a verem belsejében lévő elemeket is. Az elemek indexszel rendelkeznek, melynek segítségével hivatkozhatunk rájuk:

Változónév.Elem(*indexkifejezés*)

A veremben lévő elemek értékét azonban általában nem módosíthatjuk, tehát a hivatkozás nem állhat egy értékadó utasítás bal oldalán.

Az *ElementAt* függvénynév elhagyható: *Verem.ElementAt*(3) helyett: *Verem*(3). Az elem azonban így sem módosítható.

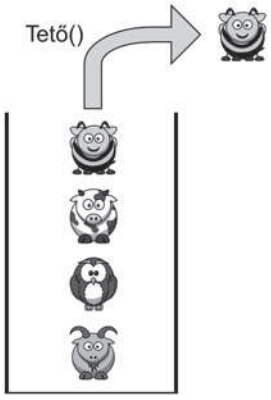
A verem tartalmát a *Töröl* metódussal töröljük:

Változónév.Töröl()

Törlés után a verem üres lesz (elemszáma nulla).

Az egyes programozási nyelvek további metódusokkal segítik a verem kezelését. Ezek felsorolását a *Programozási összefoglalóban* találjuk.

2. gyakorlat. A *Verem* kódfájl alapján tekintsük át a veremműveletek használatát!



A Tető metódus nem módosítja a verem tartalmát

A verem alkalmazása

Írjunk programot, amely a 10-es számrendszerből más számrendszerbe vált át egy pozitív egész számot! Ehhez a számot maradékosan osztjuk az új alapszámmal mindaddig, amíg a hányados 0 nem lesz. Ekkor a maradékok fordított sorrendben adják meg az új számrendszerben felírt értéket.

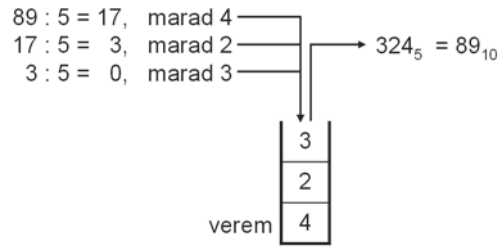
A fordított sorrendet egy verem segítségével állítjuk vissza a szokásos, balról jobbra csökkenő helyiérték szerinti elrendezésbe. Az algoritmus a deklarációktól eltekintve:

```
Be: Szám, ÚjAlap
CIKLUS
    Hányados = Szám Div ÚjAlap ' a maradékos osztás hányadosa
    Maradék = Szám Mod ÚjAlap ' a maradékos osztás maradéka
    Szám = Hányados
    Verem.Verembe(Maradék)
AMÍG Hányados > 0
CIKLUS VÉGE
CIKLUS AMÍG Verem.Elemszám > 0
    Ki: Verem.Veremből()
CIKLUS VÉGE
```

A *Maradék* változót elhagyhatjuk, ha a maradékot egyből betesszük a verembe:

```
Verem.Verembe(Szám Mod Újalap)
```

Az utasítások átrendezésével a *Hányados* változóra sincs szükségünk (lásd az alábbi gyakorlat megoldását).

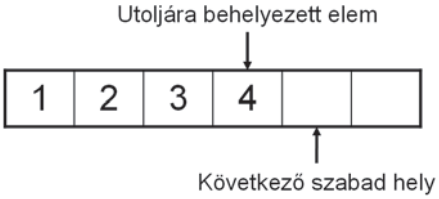


Számrendszerváltás verem segítségével

3. gyakorlat. Írjuk meg az átszámítást végző programot! Gondoskodjuk arról, hogy a 10-nél nagyobb alap esetén az ábécé betűi jelenjenek meg a számjegyek helyett (A=10, B=11 stb.)! Küszöböljük ki az algoritmusból a *Hányados* változót!

Feladatok

1. Írjunk eljárásokat az alábbi feladatok végrehajtására! Az eljárások paramétereit válasszuk meg célszerűen!
   a) Töröljük a verem második elemét (az első elem maradjon a verem tetején)!
   b) Töröljük a verem *n*-edik elemét! Az *n* legyen az eljárás paramétere.
   c) Töröljük a verem utolsó (legalul lévő) elemét! A verem tartalma egyébként maradjon változatlan!
2. Úgy is oldjuk meg a feladatokat, hogy törlés helyett egy új elemet helyezünk a verem megadott helyére!
3. Készítsünk programot, amely két, az elemeket nagyság szerint rendezve tartalmazó vermet egyesít egy új verembe úgy, hogy megmaradjon az elemek rendezettsége! Írjuk meg a programot úgy, hogy
   a) az eredeti vermekre már nincs szükség;
   b) a végrehajtás után az eredeti vermek tartalma maradjon változatlan!
4. Írjunk programot, amely beolvas egy aritmetikai kifejezést! A program ellenőrizze a zárójelezés helyességét! Azaz szabályos-e a zárójelek egymásba ágyazása és párosítása?
5. Bővítsük az előző programot! A kerek zárójelek mellett engedje meg a szögletes és kapcsos zárójelpárok használatát is!
6. Írjunk programot, amely beolvas egy karaktersorozatot, majd verem segítségével ellenőrzi, hogy szimmetrikus-e a sorozat (azaz visszafelé olvasva ugyanazt a sztringet kapjuk-e)!



Verem megvalósítása tömbbel

7. Készítsünk programot, amelyben tömb segítségével szimuláljuk egy verem működését! Írjuk meg a verem tulajdonságait és metódusait előállító függvényeket, illetve eljárásokat!
8. Deklaráljunk struktúrát (rekordot) a *Diákok.txt* fájl adatainak feldolgozásához! Olvassuk be az adatokat egy verembe! Két további verembe válogassuk szét a fiúkat és a lányokat! Az eredetihez képest milyen sorrendben helyezkednek el ez utóbbi vermekben a diákok?
9. Verem segítségével párosítsuk össze a *Diákok.txt* fájlban lévő fiúkat és lányokat! Az egymás után beolvasott, azonos nemű diákokat addig helyezzük a verembe, amíg ellenkező nemű diák nem érkezik! Az ő párja a verem tetején lévő diák lesz. Ezt a folyamatot ismételjük, amíg be nem olvassuk az összes diákot! Mikor marad diák a veremben?
10. Helyezzünk el egy veremben véletlenszámként megválasztott elemszámú, véletlenszámokkal feltöltött tömböket! Másoljuk át egy nagy tömbbe a veremben lévő tömbök elemeit!