Dr. Holló Csaba, SZTE TTIK, Szoftverfejlesztés Tanszék

Programozási alapismeretek kurzus

Olvasási idő:
8 perc

Alapvető programozásnyelvi elemek C-ben és PHP-ban

VI. Pufferek használata

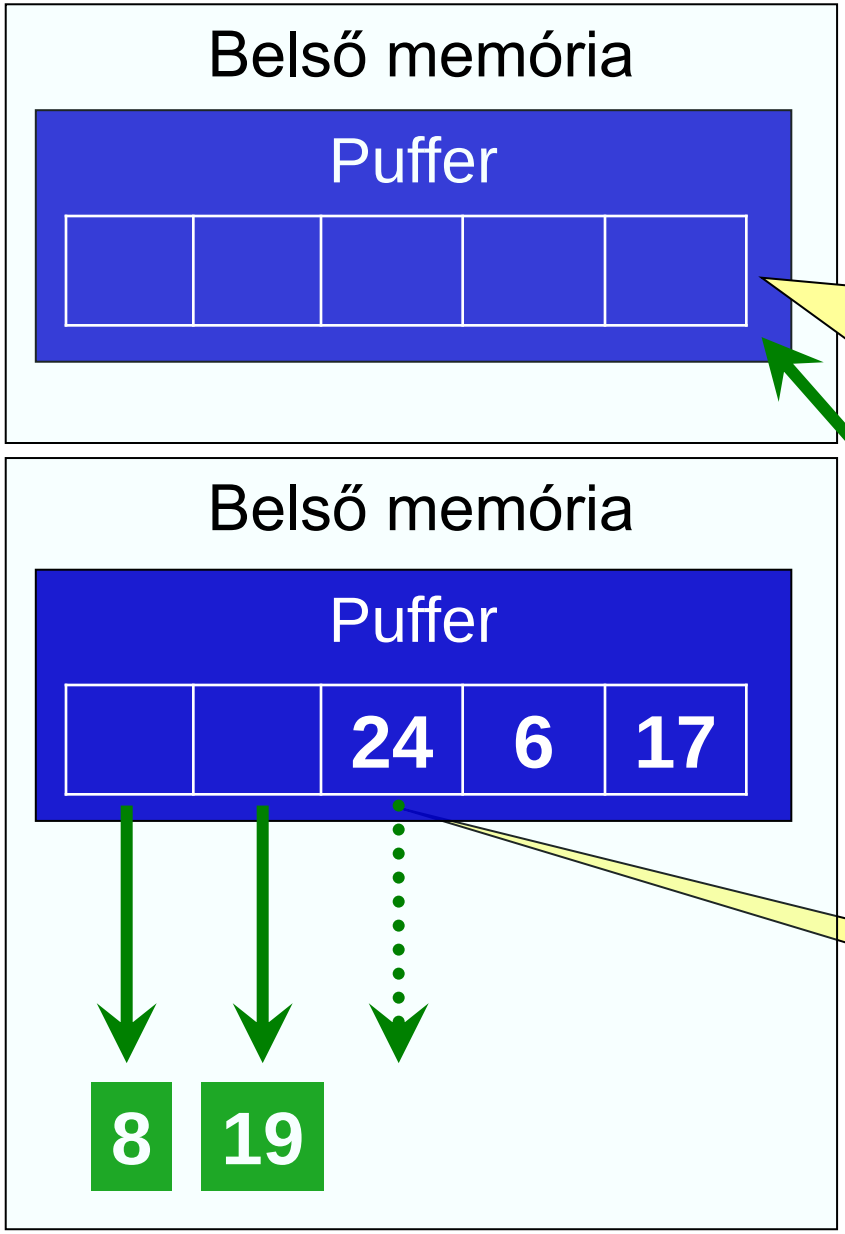
A lecke célja A pufferelés céljának és működésének jobb megértése.

Tudás Ismeri a puffer fogalmát és használatának következményeit.

Képesség A programok hatékonyságának növelése érdekében puffereket használ, figyelembe véve azok mellékhatásait.

Puffer C-ben

- Egy olyan tároló, ahol ideiglenesen tárolunk valamit, és amit a folyamataink hatékonyságának növelésére használunk.
- Fájlból olvasáskor alapesetben minden adatért elmegyünk és kiolvassuk. De az adatok fájlból való olvasása, különösen a hagyományos merevlemezek esetén, sokkal lassúbb folyamat, mintha azokat a belső memóriából olvasnánk.
- Hatékonyabb, hogyha elkülönítünk egy memóriaterületet, ezt nevezzük **puffernek**, és egyetlen olvasással nem csak a kért adatot hozzuk be, hanem az utána következő néhányat is.
- Hogyha a következő olvasáshoz nem a pufferben levő adat kell, vagy a puffer kiürült, akkor természetesen újból el kell menni a merevlemezre olvasni.



A pufferből próbálunk olvasni. Hogyha nincs adat a pufferben, vagy nem az ott levő (következő) adatot szeretnénk olvasni, akkor a merevlemezről egyszerre több adatot beolvasunk a pufferbe.

Forrás: https://pixabay.com/hu/photos/hdd-merevlemez-meghajt%C3%B3-makr%C3%B3-453775/

Hogyha van megfelelő adat a pufferben, akkor onnan olvasunk.

Példa

```
...; char t[10]; int i1, i2;
printf("1. szam = ");
scanf("%d",&i1);
printf("i1 = %d\n",i1);
printf("2. szam = ");
scanf("%d",&i2);
printf("\n i2 = %d",i2);
printf("\n Szoveg = ");
scanf("%s",t);
printf("\n t = %s", t); …
```

Puffer

1. szam =

7 vagy 8

i1 = 7

vagy 8

2. szam =

"vagy"-ot nem tudja egész értékként értelmezni → i2-be nem olvas be semmit, szemét marad benne

i2 = 104 → 

Szoveg =

t = vagy

Példa

- Az előző példában bekértünk két egész értéket és egy stringet.
- Tegyük fel, hogy az első érték kérésekor a bizonytalan felhasználó azt írja be, hogy „7 vagy 8”.
- A program az első egész értéket (7) beolvassa, de a 2. számnak nem fog értéket kérni a felhasználótól, és az i2 változóba nem olvas be semmit (így annak kiírt értéke memóriaszemét lesz). Ez további egész értékek kérésekor is így történne, mindaddig, amíg a pufferből a stringet el nem távolítjuk, jelen példában azzal, hogy beolvassuk.
- Hasonló esetekben arra lehet szükség, hogy a program adott pontján a puffert teljesen kiürítsük. Ezt megtehetjük a `while((getchar()) != '\n');` utasítással, vagy nem szabványosan az fflush(stdin) függvényhívással.

Puffer C-ben

- Ugyanez a jelenség van akkor is, amikor a merevlemezre írni szeretnénk: alapvetően a pufferbe írunk, melynek a tartalma akkor íródik ki ténylegesen a merevlemezre, amikor a puffer megtelt, vagy ezt kifejezetten kérjük az `fflush(stdout);` függvényhívással, vagy a program véget ér.
- Megjegyezzük, hogy az I/O műveletek nem mindig történnek pufferen keresztül, a puffer használata a C nyelv implementációjától, illetve a használt utasítástól vagy függvénytől függ.

❖ További információkat a pufferelés használatáról C-ben Benkő Tiborné, Benkő László, Tóth Bertalan: **Programozzunk C nyelven** c. könyvében találunk.

Puffer PHP-ben

- Itt is tipikusan arra használjuk, hogy kiíráskor ideiglenesen visszatartsunk kiírandó adatokat. Ez történhet például azért, mert szabvány szerint normál adatok elküldése után nem küldhetnénk bizonyos speciális információkat.
- Egy erre szolgáló függvény meghívásával előírhatjuk azt, hogy egy adott másik függvény meghívásáig minden olyan adat, amit látszólag a felhasználónak írunk ki, egy pufferbe kerüljön.

Puffer PHP-ben

- Megadhatjuk azt is, hogy a pufferelés befejezésekor, annak tartalmának elküldése előtt, milyen módosítások történjenek a puffer tartalmában (pl. kicserélünk bizonyos tartalmakat).
- Pufferelés közben elkezdhetünk újabb pufferelést, akkor annak befejezésekor az elküldendő tartalom az őt beágyazó pufferbe kerül bele, és ezeken az adatokon is megtörténnek a beágyazó puffert lezáró módosítások.

❖ További információk a PHP pufferekről.

Kérdések, feladatok

1. C-ben hogyan tudjuk csökkenteni a pufferelléssel összefüggő hibák esélyét?
2. Módosítsuk a fenti C példaprogramot a leírt pufferürítési technikákkal úgy, hogy a program az elvártaknak megfelelően működjön!
3. Írjunk egy programot, melyben beolvasunk egy stringet és közvetlenül utána egy karaktert is, majd ezeket írassuk ki.

   Segítség. Olvassuk el a puffer kiürítésénél linkelt szakirodalmat.
4. * Mondjunk példát olyan esetre, amikor PHP-ban a pufferelés hasznos lehet?

   Segítség. Olvassuk el a következő oldalt.

EFOP-3.4.3-16-2016-00014

AP1 HALLGATÓI DIPLOMA-SZERZÉST SEGÍTŐ SZOLGÁLTATÁSOK

Jelen tananyag a
Szegedi Tudományegyetemen
készült az
Európai Unió támogatásával.
Projekt azonosító:
EFOP-3.4.3-16-2016-00014