

## **Tudnivalók az otthon kidolgozandó feladatokról**

- Otthon kidolgozandó feladat megoldásának beküldése csak azok számára kötelező, akik fölvtették az Assembly programozás konzultáció kurzust. Minden hallgató, aki az általa választott vagy a számára kiosztott otthon kidolgozandó feladat megoldását beküldi, a megoldás minőségétől függően minimum 0, maximum 10 pontot kap.
- A feladat választani az ETR Coospace moduljának segítségével lehet a <http://www.coosp.etr.u-szeged.hu> címen **2015. március 16. 20:00-tól**. Bejelentkezni ugyanazzal a felhasználónévvel és jelszóval kell, mint amellyel az ETR-be. Bejelentkezés után az „IB676g-XX Assembly programozás gyakorlat” színteret választva a „Jelentkezés” fejléccel ellátott „Otthon kidolgozandó feladatok – jelentkezés” linkre kattintva érhető el az a felület, melynek segítségével kiválasztható a megoldani kívánt feladat. A választás határideje **2015. április 26. 23:55**.
- Egy programot maximum 2 hallgató választhat gyakorlati csoportonként. A feladatok kiosztása a választás sorrendjében történik automatikusan. Ha már mindkét hely elkelt egy adott feladatra, akkor a Coospace már nem enged több jelentkezést.
- Az otthon kidolgozandó feladat programját Intel 8086/8088/80x86 assembly nyelven kell elkészíteni úgy, hogy a kabinetben lévő 16-bites MASM assemblerrel, vagy a gyakorlaton megismert módon 32-bites környezetben Visual Studioban fordítható legyen.
- Hallgatók közötti kooperáció, külső források (internet, szakirodalom) felhasználása megengedett, de a beadott végeredményt a hallgatónak ismernie kell. Nem elfogadható az a program, amelynek a működését a készítője nem ismeri, nem tudja elmagyarázni annak főbb részeit!
- Az otthon kidolgozandó feladat beküldési határideje: **2015. május 3. (vasárnap) 23:55**. A feladatokat szintén a Coospace-en kell majd beadni az „Feladatok” fejléccel ellátott „Otthon kidolgozandó feladat” linken. A Coospace lehetővé teszi a több alkalommal történő feltöltést is, de a gyakorlatvezetők mindig csak az **utoljára feltöltött** fájlt tudják figyelembe venni!
- A választható programok pontos működése helyenként nem teljesen van megfogalmazva. Amennyiben valami nincs konkrétan kikötve, bármilyen

technikai megoldás alkalmazható, lényeg hogy a program a várt funkcionalitást biztosítsa.

- A program kódjának tartalmaznia kell a hallgató nevét, csoportját és EHA kódját komment formájában. Szintén komment formájában tartalmaznia kell a program input/output specifikációját (azaz hogy pontosan hogyan használható) és rövid, pár soros leírását.
- **Értékelés:** a program elfogadásának alapfeltétele, hogy fordítani és futtatni lehessen a kabinetes környezetben MASM assemblerrel Dosbox környezetben, vagy a lenti linkeken található leírás alapján a Visual Studio-ban. Amennyiben ez nem teljesíthető, az otthon kidolgozandó feladat – és amennyiben a hallgató fölvette az Assembly programozás konzultációt, a kurzus teljesítése is sikertelen. **A programot az utolsó gyakorlat alkalmával kell bemutatni.** A bemutatás során demonstrálni kell a programot futás közben, és röviden el kell tudni mondani annak működését, főbb részeit, szükség szerint a program megvalósításával kapcsolatos kérdésekre válaszolni.
- Kötelező program bemutatás utáni javítására, határidőn túli pótlására nincs lehetőség.

### **Leírás a programok Visual Studioban történő 32-bites fordításáról:**

- VS 2010 alatt: <http://www.kipirvine.com/asm/gettingStartedVS2010/index.htm>
- VS 2012 alatt: <http://www.kipirvine.com/asm/gettingStartedVS2012/index.htm>

# Otthoni kidolgozásra választható feladatok listája

## 1. Ceasar-kódolás (<http://hu.wikipedia.org/wiki/Caesar-rejtjel>)

- Input: egy karakter és egy karaktersorozat bekérése a billentyűzetről
- Output: az inputon kapott karaktersorozat kódolt vagy dekódolt formában kiírva a képernyőre
- Feladat részletesen: Az első karakter azt mondja meg, hogy kódolni vagy dekódolni szeretnénk az input szöveget (pl.: 1-kódolás, 2-dekódolás). Kódolás esetén végig haladunk a szövegen és minden egyes betűt az ábécében egy tőle meghatározott távolságra lévő betűvel kell helyettesíteni. Így például, ha mondjuk az eltolódás 3, az **angol** ábécében az A-t a D-vel, a B-t az E-vel stb. kell helyettesíteni. Ezt az eltolást standard módon előre megadhatjuk a programkódban. Dekódolás esetén hasonló a helyzet, csak az eltolást visszafelé végezzük. Ügyelni kell arra, hogy például, ha az eltolás 3, az angol ábécé utolsó 3 betűjét, az ábécé első 3 betűjével helyettesítsük, és fordítva, ne fussunk ki az ábécéből. (X<->A, Y<->B, Z<->C).

## 2. Egyábécés helyettesítő kódolás kulcs alapján

- Input: egy karaktersorozat bekérése a billentyűzetről
- Output: az inputon kapott karaktersorozat kódolt formában kiírva a képernyőre
- Feladat részletesen: Adott egy 26 karakter hosszú kulcs, mely az angol ábécé betűinek permutációja. Az egyábécés helyettesítő kódolás a kulcs alapján úgy történik, hogy az angol ábécé betűit a kulcs adott pozícióján lévő betűjével helyettesítjük. Tehát az angol ábécé betűit („abcdefghijklmnopqrstuvwxy”) rendre helyettesítjük a kulcsban szereplő (pl. „dkvqfibjwpescxhtmyauolrgzn”) betűkkel, vagyis, az 'a' betűket 'd' betűkre, a 'b' betűket 'k' betűkre, a 'c' betűket 'v' betűkre, és így tovább lecseréljük. A helyettesítő kulcsot megadhatjuk előre a kódban. Az egyértelműség kedvéért használjuk az itt megadott helyettesítő kulcsot. Elég csak a kisbetűket lekezelni.

## 3. Számolás megvalósítása – Összeadó gép

- Input: a felhasználó billentyűzetről maximum 20 jegyű egész számokat és műveleteket ír be. Sorban egy szám, majd egy művelet. A műveletek: + (összeadás) és – (kivonás), Enter jelzi a formula végét. A program csak az éppen következő inputnak megfelelő karaktereket engedje beírni (vagy számjegyet, vagy műveleti jelet).

- Output: a beírt számokkal a műveleteket elvégzi, és az eredményt kiírja a képernyőre. A kapott eredménnyel számoljunk tovább egészen addig, amíg a felhasználó ki nem lép.
- Megjegyzés: Legegyszerűbb, ha úgy dolgozunk, mintha papíron végeznénk a számolást.

#### **4. Számolás megvalósítása nagy számokkal (input, kiírás, megvalósítás) – Szorzás**

- Input: két, maximum 10 jegyű egész számot olvassunk be billentyűzetről és írjuk is ki a képernyőre. A számok végét Enter leütésével jelezheti a felhasználó, de 10 számjegy után automatikusan lépünk a következő számra.
- Output: ha megvan a két szám, szorozzuk őket össze és írjuk ki az eredményt a képernyőre.
- Megjegyzés: Legegyszerűbb, ha úgy dolgozunk, mintha papíron végeznénk a számolást.

#### **5. Anagramma ellenőrzés**

- Input: olvassunk be a billentyűzetről két szót. A szavak végét szóköz jelezze.
- Output: írjuk ki a két szó alá, hogy anagramma, ha az egyik szó a másik szó betűinek permutációjával előállítható, különben azt írjuk ki, hogy nem anagramma.

#### **6. Hamming kódolás**

- Olvassunk be a billentyűzetről egy négyjegyű hexadecimális számot, és írjuk ki a páros paritású Hamming kódolását binárisan!

#### **7. Bit hibajavítás**

- Olvassunk be a billentyűzetről egy 16 bites bináris szám páros paritású Hamming kódját (21 bites szám)! Állapítsuk meg, hogy hibás-e! Ha hibás, akkor írjuk alá a helyes számot, ha helyes, akkor írjuk ki, hogy helyes!

#### **8. Akasztófajáték**

- Szókitalálás játék előre megadott szókészlettel. A szavakat az adat szegmensben tároljuk, változó hosszúságú 0 értékű byte-ra végződő sztringeket használva. Induláskor a gép sorsolja ki az egyik szót és írjon ki annyi „-” jelet a képernyőre ahány betűből a szó áll. A játékos betűk lenyomásával tippelhet. Ha a szóban van olyan betű, akkor írjuk ki a megfelelő helyre a „-” jel helyére. Ha nincs, akkor növeljük a rossz tippiek számát. Ügyeljünk arra, hogy csak betűket vegyünk tippnek, és a kis/nagybetűket ne különböztessük meg! Adott számú rossz tipp után, vagy amennyiben kitalálták a szót, a program írjon ki ennek megfelelő üzenetet és lépjen ki.

## 9. Prímszámkereső

- Keressük meg és írjuk ki a képernyőre az összes prímszámot 1 és 10 000 között.

## 10. Gyorsrendezés (számok)

- Input: egy maximum 10 elemű, 32 bites előjeles egész számokat tartalmazó tömb az adatszegmensben.
- Output: készítsük el a tömb rendezett változatát a gyorsrendezés algoritmus szerint. A rendezést ne helyben végezzük el, hanem készítsünk a tömbből másolatot előtte. Az eredményt írjuk ki a képernyőre, vesszővel elválasztva.

## 11. Betű hisztogramm

- Számoljuk meg, hogy az input karaktersorozatban melyik betű hányszor fordul elő.
- Input: A klaviatúráról beolvasott legfeljebb 80 karakter hosszú karaktersorozat.
- Output: a képernyőre kiírt táblázat tartalmazza az előforduló betűket és az előfordulások számát.

## 12. Buborékrendezés (sztringek)

- Input: egy 5 elemű, max. 10 karakter hosszú sztringre mutató pointerből álló tömb az adatszegmensben + a hozzá tartozó sztringek. A sztringek hosszúsága változó, és 0 értékű byte jelzi a végüket.
- Output: készítsük el a tömb rendezett változatát a buborékrendezés algoritmus szerint. A rendezést helyben végezzük el. Az eredményt írjuk ki a képernyőre.

## 13. Törtszámok megvalósítása

- Valósítsuk meg a 4 alapműveletet valódi törtszámoláshoz. A törteket számláló/nevező alakban tároljuk, egyszerűsítve minden esetben. Egész számokat is törteként  $x/1$  alakban kezelhetjük. Elég, ha számláló és a nevező belefér egy-egy regiszterbe.
- A műveletek bemutatásához készítsünk kis programot, ami 1-1 példaszámolást végez mindegyik műveletre. Az eredményt számláló/nevező alakban kell kiírni.

## 14. Számátváltó 2-16 rendszerek között, oda-vissza

- Készítsünk számrendszerek közötti átváltó programot.
- Input: számrendszer típusa (2-16 közötti szám), majd egy szám beírása billentyűzetről és végül újból egy számrendszer típusa. Az adatok beírását Enter zárja.
- Output: az elsőnek megadott számrendszerből váltsuk át a beírt számot a másodikba. Ügyeljünk arra, hogy csak az elsőnek beírt számrendszernek

megfelelő számjegyeket engedjünk beírni! Az eredményt írjuk ki a képernyőre.

### **15.Tökéletes számok**

- Keressük meg és írjuk ki a képernyőre az összes szóban (16 biten) elérő tökéletes számot!

### **16.Barátságos számok**

- Keressük meg, és írjuk ki a képernyőre az összes szóban (16 biten) elérő barátságos számpárt!

### **17.Szó ismétlések száma**

- Az input szavait szóközök választják el egymástól. Meg kell számolni, hogy hányszor fordul elő az, hogy egy szó megismétlődik az inputban.
- Input: A klaviatúráról beolvasott legfeljebb 80 karakter hosszú karakter sorozat.
- Output: a szó ismétlések száma.

### **18.Mértani sorozat**

- Állapítsuk meg, hogy egy számsorozat mértani sorozatot alkot-e.
- Input: a billentyűzetről beolvasott előjel nélküli számok sorozata.
- Output: "Mértani sorozat", ha a számok mértani sorozatot alkotnak. "Nem mértani sorozat" egyébként.

### **19.Rész-string keresés**

- Input: A klaviatúráról beolvasott két, legfeljebb 80 karakter hosszú karaktersorozat.
- Output: „Nincs egyezés.” kiírása a képernyőre, ha az első karaktersorozat nem tartalmazza a másodikat, különben egy egész szám, az a pozíció ahonnan kezdve az első sorozatban szerepel a második.

### **20.Hanoi tornyai**

- Input: A klaviatúráról beolvasott három egész szám, az első 1 és 10, a többi 1 és 3 közötti.
- Output: A Hanoi tornyai játék képernyőre kiírt megoldása feltételezve, hogy az első szám a torony magasságát jelöli, a másik kettő pedig a kezdő és a cél pozíciókat.

### **21.Növekvő részsorozat hossza**

- Határozzuk meg egy számsorozat leghosszabb növekvő részsorozatának a hosszát.
- Input: számsorozat.
- Output: a leghosszabb növekvő részsorozat hossza.