Programozás alapjai 1. (BMEVIEEA100)

Gyakorlat anyaga a 4. oktatási héten

A 4. héten elmaradnak a pénteki órák a Schönherz kupa miatt, így a pénteki csoportokban az 5. héten kerül terítékre a 4. heti anyag. A pénteki csoportokban már a 3. héten adjuk fel az „Azonosítók, operátorok” című fejezetet a tesztgyűjteményből. Végleges teszt csak szeptember végére lesz, a hallgatók csak utána nyomtassák ki a feladatokat! Az első kis zh-t a 6. héten írják, melynek anyaga a Bevezetés, valamint az Azonosítók és operátorok fejezet, utóbbi megoldásait pedig az 5. héten kell átnézni.

Íme a félév első felének tervezett időbeosztása, hogy jobban átláthassátok a dolgokat. Fekete háttér elmarad, sárga háttér: az előző szombatra áthelyezett péntek. Az A, B, C betűk a táblázat alatt látható fejezetcímek a feladatgyűjteményből.

Hét	Hétfő			Szerda			Péntek		
	Anyag	HF	ZH	Anyag	HF	ZH	Anyag	HF	ZH
2	Bevezetés	A		Bevezetés	A		Bevezetés	A	
3	Adattípusok			Adattípusok			Adattípusok	B	
4	Operátorok	B		Operátorok	B				
5	Állapotgép, többdimenziós tömbök	C, D		Állapotgép, többdimenziós tömbök	C, D		Operátorok	C	
6	Pointerek, dinamikus tömbök	E	A, B	Pointerek, dinamikus tömbök	E	A, B	Állapotgép, többdimenziós tömbök	D	A, B
7	Stringek, gyakorlás	F		Stringek, gyakorlás	F		Pointerek, dinamikus tömbök, stringek	E, F	
8	Rendezés, keresés, hashing	G	C, D, E	Rendezés, keresés, hashing	G	C, D, E	Rendezés, keresés, hashing	G	C, D, E

A Bevezetés
B Azonosítók, operátorok
C Adatbevitel billentyűzetről, kiírás képernyőre
D A C nyelv utasításai
E Függvények használata
F Enum, tömb, struktúra, stringek
G Keresés, rendezés, hashing
H Dinamikus adatszerkezetek
I Függvényérték paramétersoron, függvénypointer
J Fájlkezelés

3. gyakorlat

Az óra első felében a házi feladatot beszéljük meg. A példatár elérhetősége: http://www.eet.bme.hu/~szvdom/peldatar/

A példák a következők:

2. fejezet:

5. Készítsen programot, amely bekér a felhasználótól két valós számot, és az összegét kiírja a képernyőre!
Nehézség: 1

6. Készítsen programot, amely bekér a felhasználótól egy valós számot (Celsius fok), az eredményt átváltja Fahrenheit értékbe, és kiírja az eredményt a képernyőre (0°C=32°F, 40°C=104°F, lineáris)!
Nehézség: 1

47. Készítsen programot, mely két időpontot kérdez a felhasználótól (óra, perc, másodperc külön), majd kiszámítja a két időpont közötti időtartamot másodpercben, és az eredményt kiírja a képernyőre.
Nehézség: 2

49. Készítsen programot, mely bekéri a felhasználótól, hogy a kasszában hány 100, 200 és 500 Ft-os található. A program számolja ki, hogy mennyi a beírt pénz összege Forintban.
Nehézség: 2

50. Készítsen programot, mely kiszámítja az első N természetes szám
a.) összegét: 1+2+3+...+N
b.) szorzatát: 1*2*3*...*N
Nehézség: 1

51. Készítsen programot, mel bekér egy számot (k), majd kiszámítja az alábbi összeget:
y=1x2 + 2x3 + 3x4 + ... + k(k+1)
Nehézség: 2

61. Írjon programot, mely kiszámítja az

```
a(n)=(1+ 1/n)^n
```

sorozat k-adik elemét. A k változó értékét kérdezze meg a felhasználótól! (A hatványozáshoz nem használhatja a pow függvényt.)
Nehézség: 2

84. Készítsen programot, mely a felhasználótól beolvasott természetes számot visszafele írja ki. Például 651-re a válasz: 156.
Nehézség: 2

95. Készítsen programot, mely a felhasználótól három síkbeli pont (x,y) koordinátáinak bekérése után
a.) meghatározza, hogy a pontok köré írható-e kör
b.) megadja a köréírható kör sugarát és középpontjainak (x,y) koordinátáit.
Nehézség: 3

96. Készítsen programot, mely a felhasználótól négy térbeli pont (x,y,z) koordinátáinak bekérése után
a.) meghatározza, hogy a pontok köré írható-e gömb
b.) megadja a köréírható gömb sugarát és középpontjainak (x,y,z) koordinátáit.
Nehézség: 4

117. Készíts programot, amelyik megadja, hogy valahány forintot hogy lehet a forgalomban lévő címletű pénzekből legkevesebb számúval kifizetni. Például: 1040 Ft = 1 db 1000-es, 2 db 20-as.
Nehézség: 2

121. Készítsen programot, amely egy másodfokú polinom (a*x*x+b*x+c) gyökeit adja meg; beleértve azt az esetet, ha azok komplexek!
Nehézség: 1

148. Készítsen programot, amely egy felhasználó által megadott számot kiír az ugyancsak általa megadott számrendszerben. Például a 16 a 3-as számrendszerben: 1*3^2 + 2*3^1 + 1*3^0.
Nehézség: 2

160. Készítsen programot, mely egy természetes szám számjegyeinek összegét képes meghatározni.
Nehézség: 2

11. fejezet:

2. Készítsen programot, amely bekéri a felhasználó nevét, majd üdvözli őt a nevén szólítva!
Nehézség: 1

27. Készítsen függvényt (myToUpper) mely egy sztringben képes a latin abc betűit nagybetűssé alakítani! A bemenet legyen a sztringre mutató kezdőpointer, a végeredmény ugyanebbe a sztringbe kerüljön bele!
Nehézség: 1

43. Készítsen programot, mely bekér egy keresztnevet, majd azt betűnként függőlegesen lefelé kiírja. Például ha a név "Imre", akkor az eredmény:

```
I
m
r
e
```

Nehézség: 1

45. Készítsen programot, mely bekér egy mondatot, majd
a.) megszámolja és kiírja, hogy a mondatban hány szóköz található.
b.) kiírja a mondatot szóközök nélkül.
Nehézség: 2

76. Készítsen függvényt, mely paraméterben egy sztringet és további két karaktert (mit és mire) kap. A függvény keresse meg a sztringben a "mit" változóban megadott karaktereket, és cserélje azokat a "mire" változóban megadottakra. A függvény visszatérési értéke a kicserélt karakterek számát jelentse.
Nehézség: 2

78. Készítsen programot, mely egy, a felhasználó által megadott 1 és 99.999 közötti természetes számot képes kiírni betűvel! 2000-ig minden számot egybeírunk, 2000 fölött az ezres és ezer alatti rész közé kötőjelet kell tenni. Példák:

```
625: hatszazhuszonot
44: negyvennegy
1975: ezerkilencszazhetvenot
8000: nyolcezer
23470: huszonharomezer-nyolcszazhetven
```

Nehézség: 4

85. Készítsen függvényt (numLower), ami megkap egy stringre mutató pointert, és visszaadja az adott szövegben található kisbetűk számát. (Használhatja a ctype.h islower(int) függvényét.)
Nehézség: 2

90. Készítsen programot, mely adott sztringben megszámolja, hányszor fordul elő az "a" névelő. A névelő lehet mondat elején, de végén nem, viszont vessző állhat előtte is és utána is, egyébként szóköz karakterek határolják.
Nehézség: 3

138. Készítsen programot, amelyik egy megadott jelszóról eldönti, hogy az kellően erős-e. Erős jelszó az, amelyik tartalmaz kisbetűt, nagybetűt és számot is. Használhatja a ctype.h függvényeit.
Nehézség: 2

153. Készítsen programot, amely a felhasználó által beírt mondatot madárnyelven ír ki: "Teve tuvudsz ivigy beveszevelnivi?"
Nehézség: 2

Új anyag – operátorok

```
#define MAX_KAR 100

//*****************************************************
void operatorok(){
//*****************************************************
	int a=3,b=6,n;
	char c[MAX_KAR];

	// bitenkénti operatorok

	// and, or, xor, not => 2, 7, 5, -4
	printf("%d %d %d %d\n",a&b,a|b,a^b,~a);

	// logikai operatorok

	// and, or, not => 1, 1, 0
	printf("%d %d %d\n",a&&b,a||b,!a);

	printf("\nKerek egy egesz szamot: ");
	if(scanf("%d",&n)!=1){printf("\nHibas adat!\n");return;}
	n%2==0||printf("nem ");
	printf("paros szam\n");

	// bitek eltolása (shift)

	// 6 12 24
	printf("%d %d %d\n",a<<1,a<<2,a<<3);
	// 3 1 0
	printf("%d %d %d\n",b>>1,b>>2,b>>3);

	a<<=1;//a*=2
	b>>=1;//a/=2

	// Hany bites az int?

	for(n=0,b=1;b;b<<=1,n++);
	printf("%d bites az int\n",n);

	// egész szám binárissá konvertálása

	printf("Kerek egy egesz szamot!\n");
	if(scanf("%d",&a)!=1){printf("\nSikertelen beolvasas\n");return;}

	n=n<MAX_KAR?n:MAX_KAR;

	printf("Binaris formaban:");
	b=0;
	do{
		c[b++]=(a&1)?'1':'0';
		a=a>>1;
	}while(a!=0&&b<n);
	while(b--)putchar(c[b]);
	printf("\n");

	// egész szám binárissá konvertálása - tömb nélkül

	printf("Kerek egy egesz szamot!\n");
	if(scanf("%d",&a)!=1){printf("\nSikertelen beolvasas\n");return;}

	for(n=0,b=a; b!=0; b>>=1,n++);
	if(n!=0)for(b=0;b<n;b++)putchar(a&(1<<(n-b-1))?'1':'0');
	else putchar('0');
}
```

A #define-nal preprocesszor konstanst definiálunk, így ha később meg akarjuk változtatni a tömb méretét és a tömb méretéhet igazodó utasításokat, akkor elég csak egy helyen megváltoztatni a kódot. Használhatunk helyette const int MAX_CAR-t is, ekkor nem az előfeldolgozó, hanem a fordító kezeli a konstanst.

Bitenkénti operátorok: négyféle, egész számok kettes számrendszerbeli alakjával végzi a műveleteket. Írjuk fel az igazságtáblákat!

Logikai operátorok: csak háromféle, nincs KIZÁRÓ VAGY (XOR). Logikai értéken végzi a műveletet. Az egész számok is logikai jelentéssel bírnak: 0=HAMIS, 1=IGAZ. Logikai művelet eredmény egész számként használható: IGAZ=1, HAMIS=0:

```
printf("%d %d %d %d\n",3||0,3&&0,3<0,3>0); // 1 0 0 1
```

Logikai operátor, mint kiértékelési pont

Az && és a || logikai operátor különleges tulajdonsággal rendelkezik: kiértékeli a tőle balra lévő kifejezést, és annak igaz vagy hamis értékétől függően vagy kiértékeli a jobb oldalát, vagy nem. Ha az && operátor hal oldalán HAMIS érték áll, akkor a jobb oldalán állhat bármi, az eredmény HAMIS lesz. Ha a || operátor bal oldalán IGAZ kifejezés áll, a jobb oldalon állhat bármi, a kifejezés értéke IGAZ lesz.

Pl:.

```
a)if(a!=0&&b/a>2)csinald(); /* ha a==0, akkor biztosan hamis az állítás,
tehát a b/a-t nem számolja ki, így nem lesz 0-val való osztás. */
b) a=3; b=a++ + (c=a); printf("%d %d %d\n",a,b,c); // 4 6 3 v. 4 7 4
c) a=3; if(a++==3&&b==(c=a))printf("%d %d %d\n",a,b,c); // 4 4 4
```

Az első esetben az ’a++’ értéke ’a’ eredeti értéke, a ’c=a’ részben viszont már fordítótól függ, hogy ’a’ eredeti, vagy megnövelt értéke kerül c-be, és így az összegbe, csak az a biztos, hogy a ; után ’a’ már inkrementált.. A második esetben az && operátor előtt még ’a’ eredeti értéke van jelen, tehát a++==3 IGAZ lesz, és ha igaz, ki kell értékelni a jobb oldalt is, mert attól függ, hogy az egész kifejezés IGAZ vagy HAMIS. Mivel az && operátor előtt kiértékelődött a kifejezés, a léptetés is megtörtént, azaz az &&-től jobbra a értéke már 4! Tehát c-be ezért kerül 4. (Vajon miért ír ki 4-et b-re a printf, ha a b értékét sehol nem állítottuk be? Mert különben HAMIS lenne az állítás, és a printf meg sem hívódna. Ha tehát b!=4, akkor semmi sem íródik ki.)

A példafüggvényben szereplő `n%2==0||printf("nem ");` sor egy érdekes trükk: printf csak akkor hívódik, ha n%2==0 HAMIS, printf azért lehet a || jobb oldalán, mert egész értéket visszaadó függvény (hány karaktert írt ki).

Biteltolás: nem igényel magyarázatot. Bitszámlálás ennek segítségével.

Az egész szám binárissá konvertálása két verzióban is szerepel, első egy ügyetlenebb tömbös, második egy jobb => kerüljük a tömbök használatát, ha lehet.

Példa: Írjon programot, mely bekér egy pozitív egész számot, és eldönti, hogy tökéletes szám-e, azaz megegyezik-e az osztói összegével. Az osztók közé soroljuk az 1-et, de önmagát nem. Pl.: 6=1+2+3, 28=1+2+4+7+14. Ne használjon tömböt, az osztókat nem kell külön-külön tárolni, csak az összegüket.

```
#include <stdio.h>
void main(){
    int a,b,n;
    for(b=1,n=0;b<=a/2;b++)if(a%b==0)n+=b;
    printf("%s\n",a==n?"Tokeletes.":"Nem tokeletes.");
```

```
}
```

Példa:
Mennyi 157&92? 157|92? 157^92? ~157

157= 10011101b
92= 1011100b

```
 10011101  10011101  10011101
&01011100 |01011100 ^01011100
--------- --------- ---------
 00011100  11011101  11000001
 =28       =221      =193
```

(Figyeljük meg, hogy A|B=(A&B)+(A^B)!)

Házi feladat

A tesztek közül az Azonosítók, operátorok fejezet tartalma (most nem tudom idemásolni, de október 1-én már a végleges lesz fenn a weben, az akkori tartalomnak megfelelő. Az adatbázis szeptember 26. és október 1. között folyamatosan változik.

A mai megoldandó házi feladatok a következő oldalon láthatók, ismételten a Szente-Varga féle példatárból. Javaslom fénymásolva kiosztani. Ha az óra végén marad idő, természetesen belekezdhetünk már ott a megoldásba.

4. Bitműveletek, bináris számok

22. Készítsen programot, mely egy pozitív egész szám kettes számrendszerbeli alakját írja ki a képernyőre! A program kisebb szám esetén egészítse ki a kapott eredményt kezdő nullákkal úgy, hogy annak hossza 16 digit legyen (16 bites ábrázolás).
Nehézség: 1

40. Készítsen programot, mely elvégzi az alábbi műveletet, és a végeredményt kiírja a képernyőre hexadecimális alakban! A két kezdeti szám hexadecimális formában legyen adott a program számára.

```
   1011 1110
&  1111 1100
-------------
~
-------------

|  1100 1000
-------------

<<2
-------------
```

Nehézség: 1

115. Készítsen programot, mely elkészíti az alábbi logikai függvények igazságtáblázatát, és kiírja a képernyőre:
a.) ÉS
b.) VAGY
c.) NOT
d.) NOR ~(a|b)
e.) XOR
Nehézség: 1

140. Készíts programot, amelyik meghatározza, hogy az azt futtató gépen hány bites az unsigned int típus. (De ne less!)
Nehézség: 1

141. Készítsen függvényt, amelyik egy 32 bites előjel nélküli szám byte-sorrendjét megfordítja. (Tételezzük fel, hogy az unsigned int a futtató gépen 32 bites.) Például, ha a bemenet 0x11223344, a függvény kimenete legyen 0x44332211.
Nehézség: 1

41. Készítsen programot, mely elvégzi az alábbi műveletet, és a végeredményt hexadecimális alakban kiírja a képernyőre! A két kezdeti szám hexadecimális alakban legyen adott a program számára.

```
   1110 1001
&  0101 1101
-------------

|  1100 1100
-------------
~
-------------

^  1001 1010
-------------

&  1101 0001
-------------
~
-------------

^  1010 1110
-------------

|  0001 0110
-------------
~
-------------
>>1
-------------
```

Nehézség: 2

146. Eratoszthenész szitája prímszámokat keres. A módszer a következő. Felírjuk az összes számot N-ig. 2 prímszám, kihúzzuk a többszöröseit. 3 a következő, kihúzzuk n*3-at. 4-et már kihúztuk (2*2). 5 a következő prím, kihúzzuk n*5-öt stb.
Sajnos a rendelkezésre álló memória véges, ezért minden bitet ki kell használnunk Ha egy 8 bites char (tegyük fel, hogy ekkora) típusban minden bitet külön megjelölünk, egy 60000 karakteres tömböt lefoglalva 480000-ig ki tudjuk írni az összes prímszámot.
Valósítsa meg a leírt programot!
Nehézség: 4

A 146-os csak ajánlott.

3. Ciklusok

12. Készítsen programot, mely egy nxn-es mátrix formájában kiírja a pozitív egész számokat 1-től n^2-ig. Az n értékét induláskor kérje be a felhasználótól!
Nehézség: 2

13. Írjon programot, mely kiírja a képernyőre az első n Fibonacci számot. Az n változó értékét a felhasználó adhassa meg! (A fibonacci sorozat első két eleme 1, a többi pedig az előző kettő összege, azaz 1 1 2 3 5 8 13 21…
Nehézség: 2

15. Készítsen programot, mely a felhasználó által megadott x és y számok függvényében a szöveges képernyőre kirajzol egy
a.) x*y karakter méretű, "o" betűkből álló keretet!
b.) x vízszintes élhosszúságú, y magas paralelogrammát "o" betűkből. Például:

```
x=7, y=3 esetén:
 ooooooo
 ooooooo
ooooooo
```

Nehézség: 2

17. Készítsen programot, mely két szám legnagyobb közös osztóját számolja ki. A két számot a felhasználótól kérje be!
Nehézség: 2

50. Készítsen programot, mely kiszámítja az első N természetes szám
a.) összegét: 1+2+3+...+N
b.) szorzatát: 1*2*3*...*N
Nehézség: 1

74. Helyettesítsük az 1-9 számjegyekkel az autóbuszjegyen található lyukasztási helyeket! Írjon programot, amely kiírja az összes olyan buszjegy "kódját", amely három helyen van kilyukasztva! Törekedjen a képernyő minél jobb kihasználására! A program írja ki a kombinációk számát is.
Nehézség: 3

77. Készítsen programot, mely kiszámítja ex értékét a Taylor sorával. Az iteráció mélységét (a Taylor sor hosszát) a felhasználótól megadott természetes szám adja meg. A Taylor sor szerinti definíció:

$$e^x = 1 + x + (x^2)/2! + (x^3)/3! + ...$$

Nehézség: 3