Programozás alapjai 1. (BMEVIEEA100)

Gyakorlat anyaga a 3. oktatási héten

A kis zárthelyik (tesztek) tervezett ütemezése: 6., 8., 10. és 13. hét. A félév során hétfői óra nem marad el, elmarad a november 19-i gyakorlat TDK miatt, valamint két pénteki gyakorlat: két hát múlva, október 4-án Schönherz kupa, november 28. pedig nyílt nap. Ezen kívül az október 24-i pénteket az előtte való szombaton tartjuk, tehát lesz bőven kavarodás.

2. gyakorlat

Mostantól kezdve az óra eleje a házi feladat megbeszélése, a lehetőségekhez képest interaktívan. Tesztfeladatoknál mehetünk sorban végig a hallgatókon, mindenki mondja a következő feladat megoldását. Ahol szükségét érezzük, fűzzünk pár szavas magyarázatot a feladathoz. Jelenleg a múlt pénteki tesztsor van fenn a neten annyi változással, hogy a „Melyik kódolási technika helyes az alábbiak közül?” szöveg „Melyik kódolási technika javasolt az alábbiak közül?”-re változott Szeptember 26. péntek reggelig nem változik, ami most fenn van, tehát a 3. oktatási héten egységesen mindenkinek ugyanaz a feladatsor van meg. Megoldandó példa esetén hívjunk ki valakit a táblához, aki felírja a megoldását, amit aztán kijavítunk, átbeszélünk.

A múlt héten feladott tesztsor (a programba bekerült a feladatcsoport fájlba mentésének lehetősége):

```
-------------------------------------------------------------------------------
2. Egymásba ágyazhatók-e a /* */ stílusú megjegyzések a C nyelvben?
[ ] Igen.
[ ] Nem.
-------------------------------------------------------------------------------
3. A

  /* printf("/*Hello*/ Világ!\n");*/

sorban hol van a megjegyzés vége?
[ ] A Hello után, hiszen a preprocesszor buta, és mindig az első */
karakterpárost tekinti a megjegyzés végének.
[ ] A ; után, mert a preprocesszor az első nem string konstansban szereplő */
karakterpárt tekinti a /* */ megjegyzés lezárásának.
[ ] Ebben a sorban nincs vége a megjegyzésnek, mert a /* */ megjegyzés több
soros, a lezáró */ legkorábban a következő sorban lehet.
-------------------------------------------------------------------------------
4. Mire használunk megjegyzéseket a C kódban?
[ ] Arra, hogy átláthatóvá tegyük a kódunkat, magyarázatot fűzzünk a működéshez
és eltávolítsunk nem használt programrészeket.
[ ] Csak arra, hogy átláthatóvá tegyük a kódot.
[ ] Csak arra, hogy magyarázatot fűzzünk a program működéséhez.
-------------------------------------------------------------------------------
5. Okoz-e fordítási idejű hibát, ha egy C nyelvű deklarációt, definíciót, ill.
utasítást azonosítók ill. operátorok között elhelyezett sortörésekkel több
sorra bontunk szét?
[ ] Igen, mert a C nyelvben minden sorban új utasítás kezdődik.
[ ] Definíció esetében nem, deklarációt és utasítást viszont nem lehet több sorba
törni.
[ ] Nem, sőt olykor a jobb érthetőség érdekében kívánatos is a kódot több sorba
tördelni.
-------------------------------------------------------------------------------
6. Melyik kódolási technika javasolt az alábbiak közül?
A:
  while(i>n){
      printf("%d\n",i);
  }

B:
```

```
  while(i>n)
  {
      printf("%d\n",i);
  }
```
[ ] Mindkettő.
[ ] Csak az A
[ ] Csak a B

---

7. Melyik kódolási technika javasolt az alábbiak közül?
A:
```
  while(i>n){
      printf("%d\n",i);
  }
```
B:
```
  while(i>n){
  printf("%d\n",i);}
```
[ ] Mindkettő.
[ ] Csak az A
[ ] Csak a B

---

8. Melyik kódolási technika javasolt az alábbiak közül?
A:
```
  if(a>b)
  printf("Nyaktekerészetimellfekvenc.");
```
B:
```
  if(a>b)
      printf("Gőzpöfögészetitovalödönc.");
```
[ ] Csak az A
[ ] Csak a B
[ ] Mindkettő.

---

9. A függvény nevét, visszatérési típusát és paramétereinek típustát tartalmazó azonosító a
[ ] függvénydeklaráció.
[ ] függvénydefiníció.

---

10. A függvény nevét, visszatérési típusát, paramétereinek nevét és típusát, valamint a függvénytörzset, azaz a függvény megvalósítását tartalmazó programrész a
[ ] függvénydeklaráció.
[ ] függvénydefiníció.

---

11. Melyik állítás igaz az alábbiak közül?
[ ] A függvény definíciójának része a függvény deklarációja, azaz prototípusa.
[ ] A függvény deklarációjának, azaz prototípusának része a függvény definíciója.

---

12. Függvénydeklarációnál a paraméter változók nevének megadása
[ ] nem kötelező, de ha megadjuk, különbözhet is a definícióban megadottól.
[ ] kötelező, és meg kell egyezzen a definícióban megadottal.
[ ] nem kötelező, de ha megadjuk, meg kell egyezzen a definícióban megadottal.

---

13. Melyik kifejezés változódefiníció az alábbiak közül?
A: int a;
B: extern int a;
C: static int a;
D: int a=9;
[ ] Mind a négy változódefiníció.
[ ] Csak a D változódefiníció.
[ ] Az A, C és D egyaránt változódefiníció.

---

14. Melyik kifejezés változódeklaráció az alábbiak közül?
A: int a;
B: extern int a;
C: static int a;
D: int a=9;
[ ] Csak a B változódeklaráció.
[ ] Az A, B és C egyaránt változódeklaráció.
[ ] Csak az A változódeklaráció.

---

15. Az alábbiak közül melyik algoritmusmegadási módszerrel megadott algoritmus írható át könnyebben strukturált C nyelvű kóddá?
[ ] A struktogrammal megadott.
[ ] A folyamatábrával megadott.

---

16. Mire szolgálnak a fejléc (header) fájlok a C nyelvben?

```
[ ] Semmi szükség rájuk, az a céljuk, értelmes tartalomnak tűnve a
programozáshoz nem értő kollégáinkkal elhitessük, hogy többet dolgoztunk.
[ ] Tartalmazzák a több forrásmodulban használt függvények és változók
prototípusait és a konstansokat, így ezeket nem kell minden modulba kézzel
beírni vagy bemásolni.
[ ] Tartalmazzák a több modulban használt függvények kódját, így ezeket nem kell
minden modulba kézzel beírni vagy bemásolni.
-------------------------------------------------------------------------------
17. Az előfordító
[ ] eltávolítja a megjegyzéseket és a felesleges whitespace karaktereket a
C kódból, bemásolja a #include után megadott fájlok tartalmát a #include
sora helyére, behelyettesíti a #define-nal megadott makrókat és konstansokat
a kódba, a #if feltételes fordítás feltételét kiértékelve egyes
kódrészleteket eltávolít a kódból, és más, itt fel nem sorolt dolgokat is
végez.
[ ] félig lefordítja a kódot, így a rendes fordítónak könnyebb dolga van.
[ ] buta, ezért lehetőleg nem használjuk, helyette a modernebb utófeldolgozót
használjuk.
-------------------------------------------------------------------------------
18. Melyik állítás igaz?
[ ] Minden szabványos C programban kell legyen egy és csak egy main függvény.
[ ] Ha a szabványos C programban nincs main függvény, akkor a program végre-
hajtása az első függvény első utasításával kezdődik.
[ ] Én vagyok a legszebb.
-------------------------------------------------------------------------------
19. Sortörésekkel szétbontható-e a C nyelvben egy hosszú füzér (string), ezzel
növelve a kód átláthatóságát?
[ ] Igen, de a füzér tartalmazni fogja a sortörést és a következő sor elején lévő
további szóköz jellegű karaktereket is.
[ ] Nem.
[ ] Igen, de a programkódban a sortörés karakter előtt a füzért le kell zárni,
majd a következő sorban ismét megnyitni; a két szakaszt az előfordító auto-
matikusan összefűzi.
-------------------------------------------------------------------------------
20. Mi a szerepe a main elnevezésű függvénynek?
[ ] Kijelöli a program belépési pontját, a program futása a main függvény
indításával kezdődik meg.
[ ] Csak ebből a függvényből tudunk más függvényeket meghívni.
[ ] Semmilyen szerepe nincs, egyszerűen csak szeretjük így hívni az egyik
függvényt.
```

Kipróbáltam a teszt visszakérését, úgy nem működik, hogy mindenkinek más a válaszok sorrendje, ezért javasolt a tesztkérdések kinyomtatása és kiosztása.

Új anyag

Adattípusok bemutatása példákon.

- char, sizeof(char), nem definiált, hogy előjeles-e, ASCII kódolás => 0!='0'

Írjunk programot, amely bekér a felhasználótól egy akár pozitív, akár negatív egész számot %d-vel, majd kiírja %c-vel, hogy a képernyőn ugyanaz jelenjen meg.

```
//******************************************************
#include <stdio.h>
//******************************************************

//******************************************************
int main(){
//******************************************************
    int szam,elojel=1,temp,div;
    printf("Kerek egy szamot! ");
    scanf("%d",&szam); // Az & jelet ne feledjük!
    // Az összevont scanf("Kerek egy szamot! %d",&szam); nem jó, a scanf nem ír ki semmit!!!
    if(szam<0){
        elojel=-1;
        szam=-szam;
    }
    temp=szam;
    div=1;
    while(temp>0){
        div=div*10;
        temp=temp/10;
    }
```

```
    printf("%c",elojel<0?'-':'+');//?:
    if(div==1)printf("0");
    else{
        do{
            div/=10;//div=div/10
            temp=szam/div;//egész osztás
            temp+='0';
            printf("%c",temp);//nincs &
            szam%=div;//maradékképzés
        }while(div>1);
    }
    return 0;
}
```

Mit történik, ha nem számot írunk be?

Fontosabb újdonságok:

- scanf-nél &, printf-nél nincs &.
- while és do-while ciklus, mindkettő igazra működik.
- div=div*10 helyett írható div*=10.
- A ?: operátor.
- == vagy =?
- Egész osztás, hogyan lehet két egész számot úgy osztani, hogy tizedes törtet kapjunk ((double)szam/div). '0'+2='2'
- Maradékképzés operátora (%) => h=a/b, m=a%b, akkor a=h*b+m. Csak nemnegatív egészekre értelmezett.

Fordítsuk meg az elképzelést, és ezúttal karakterenként olvassuk, majd ebből gyártsunk egész számot, használjunk tömböt!

```
//*******************************************************
#include <stdio.h>
#include <ctype.h>
//*******************************************************

//*******************************************************
int main(){
//*******************************************************
    char t[50];
    int i,szam=0,elojel=1;
    printf("Kerek egy egesz szamot! ");

    i=0;
    do{
        scanf("%c",&t[i]);// ne írjuk még fel a t+i-t
        if(isspace(t[i]))t[i]=0;
    }while(t[i++]);
// scanf("%s",t);

    switch(t[0]){
        case '+': elojel=1; break;
        case '-': elojel=-1; break;
        default: if(t[0]>='0'&&t[0]<='9')szam=t[0]-'0';
                 else{printf("Nem szam\n");}
    }

    i=1;
    while(t[i]!=0){
        if(isdigit(t[i]))szam=10*szam+(t[i]-'0');
        i++;
    }
//  for(i=1;t[i]!=0;i++)
//      if(isdigit(t[i]))szam=10*szam+(t[i]-'0');

    szam*=elojel;
    printf("%d\n",szam);
    return 0;
}
```

Fontosabb újdonságok:

- Tömb: azonos típusú változóból sok, sorszám alapján választjuk ki az elemet, a kiválasztó index az első elemtől való távolság. A karakerfüzér (string) karakterek sorozata egy tömbben, melynek végét a 0 ASCII kódú karakter jelzi, ami nem azonos a ’0’ számjeggyel.
- Szöveg beolvasható karakterenként (%c), de egyszerűbb, ha stringként olvassuk (%s). Megemlíthető a gets() is, ami nem csak szóközig olvas, hanem sor végéig, de zavart okozhat, ha a scanf-fel keverjük a használatát, mert a scanf bent hagyja az újsor karaktert.
- isspace a ctype.h-ban, hozzá hasonló: isalpha, isdigit, isalnum, isupper, islower stb.
- && logikai ÉS operátor: Ha mindkét oldalán igaz állítás van, akkor lesz az eredmény igaz, egyébként hamis. Létezik logikai VAGY operátor, ahol az eredmény akkor igaz, ha az operátor legalább egyik oldalán igaz állítás szerepel. A harmadik logikai operátor a NEM, ez egyoperandusú, csak a jobb oldalán van érték, amiből ellentétest csinál, jele a ! (if(!(a>b))printf(”Nem nagyobb”);).
- A for ciklus a while ciklus tömör alakja.

Lebegőpontos számok pontossága

a) Írjuk ki, hogy két szomszédos ábrázolható lebegőpontos szám között közelítőleg hol válik a különbség 1-gyé illetve 10^{-6}-ná!

```
//******************************************************
#include <stdio.h>
//******************************************************

//******************************************************
int main(){
//******************************************************
        float f,f2;
        double d,d2;
        for(f=1.0,f2=0.0;f!=f2;f*=1.0001F,f2=f+1.0F);
        printf("Ket abrazolhato float kozott "
                "%g es %g kozott valik a kulonbseg >1-gye\n",f/1.0001F,f);
        for(d=1.0,d2=0.0;d!=d2;d*=1.0001,d2=d+1.0);
        printf("Ket abrazolhato double kozott "
                "%g es %g kozott valik a kulonbseg >1-gye\n",d/1.0001,d);
        for(f=1.0,f2=0.0;f!=f2;f*=1.0001F,f2=f+1.0e-6F);
        printf("Ket abrazolhato float kozott "
                "%g es %g kozott valik a kulonbseg >1e-6-na\n",f/1.0001F,f);
        for(d=1.0,d2=0.0;d!=d2;d*=1.0001,d2=d+1.0e-6);
        printf("Ket abrazolhato double kozott "
                 "%g es %g kozott valik a kulonbseg >1e-6-na\n",d/1.0001,d);
}
```

Eredmény:

Ket abrazolhato float kozott 1.6777e+007 es 1.67787e+007 kozott valik a kulonbseg >1-gye
Ket abrazolhato double kozott 9.00686e+015 es 9.00776e+015 kozott valik a kulonbseg >1-gye
Ket abrazolhato float kozott 16.0101 es 16.0117 kozott valik a kulonbseg >1e-6-na
Ket abrazolhato double kozott 1.71785e+010 es 1.71802e+010 kozott valik a kulonbseg >1e-6-na

Tanulság: ha egy egyenlet gyökeit keressük hat tizedesjegy pontossággal, float esetén, ha 16 fölött van gyök, nem fogjuk megtalálni, valószínűleg végtelen ciklusba jutunk, azaz lefagy a program. Ha double-t használunk, a határ 10^{10} körül van.

b) A számítási pontatlanság ellenőrzésére egy újabb példa:

```
//******************************************************
#include <stdio.h>
```

```
#include <math.h>
//*******************************************************

//*******************************************************
void main(){
//*******************************************************
        float a,b,c,x1=1.0,x2=1.0,d,y1,y2;
        int i;
        for(i=0;i<20;i++,x2*=10.0){
                a=1.0;
                b=-(x1+x2);
                c=x1*x2;
                d=b*b-4*a*c;
                y1=(-b+sqrt(d))/(2*a);
                y2=(-b-sqrt(d))/(2*a);
                printf("x1=%g\nx2=%g\ny1=%g\ny2=%g\n\n",x1,x2,y1,y2);
        }
}
```

Ez a program az $(x-1)(x-10^n)=0$ egyenlet gyökeit számolja ki. Előbb $ax^2+bx+c=0$, azaz $x^2-(1+10^n)+10^n=0$ egyenletté alakítja, majd ennek gyökeit veszi az $y_{1/2}=(-b\pm\sqrt{b^2-4ac})/2a$ képlet segítségével. Ha jól számolna, akkor a gyökök 1 és 10^n volnának, de nem ezt tapasztaljuk. n=4-nél már láthatóvá válik egy kis hiba, 1 helyett 1.00003 a gyök, és a hiba folyamatosan nő: n=8-nál már 2,00294-et kapunk 1 helyett, n=19-nél pedig 7.01818e+010-et. A 10^n gyök mindig jó. Ha a float-ot double-re cseréljük, javul a pontosság, de a hiba továbbra is fennáll: n=16-ig jó, de n=17-nél 1 helyett hirtelen 0-t kapunk, és az eredmény ennyi is marad.

A hiba oka a -b-sqrt(d) kivonás. Itt ugyanis két nagy számot vonunk ki egymásból, melyek között kicsi a különbség.

Házi feladat

Ha óra végén marad idő, már ott elkezdhetjük. Most nem adok tesztfeladatokat, majd egy hét múlva, amikor már végleges lesz a feladatsor. Ezúttal rengetek nagyon egyszerű feladatot kapnak. Mindenkinek figyelmébe ajánlom a Szente-Varga Domonkos által, Nagy Gergely segítségével összeállított példatárat, használja egészséggel laboron is, gyakorlaton is!

A példatár elérhetősége: http://www.eet.bme.hu/~szvdom/peldatar/

Innen a mai házi a 2. és 11. fejezet (Változók, aritmetika és Sztringek). Ha soknak találjuk, válogathatunk közülük, de szerintem olyan egyszerűek a példák, hogy fel lehet adni mind. Javasolt a példák fénymásolt verzióban történő kiosztása. 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

*