Programozás alapjai 1. (BMEVIEEA100)

Gyakorlat anyaga az 6. oktatási héten (4-5. gyakorlat)

A 7. oktatási hét péntekje előbbre csúszik a 6. hét szombatjára, ezért a 7. heti anyagot a szokottnál előbb kapjátok majd.

1. kis ZH

ZH írás előtt beszéljük meg a felmerülő kérdéseket, vagy ha előző alkalommal nem volt idő minden kérdés megoldását megbeszélni, akkor fejezzük be most!

A kis ZH sablonja a következő oldalon látható, gyakorlatilag megegyezik a múlt héten küldött mintával, csak nincs benne feladat, azt mindenki maga generálja a http://www.eet.bme.hu/new/index.php?option=com_docman&task=doc_download&gid=97&Itemid=94 címről letölthető csomagban található programmal. Érdemes a ZH időpontjához minél közelebb letölteni minden alkalommal, mert a hibajavítások folyamatosak. Legalább két csoport legyen, hiszen nagyon sűrűn vannak a gyakorlatokon. Mielőtt sokszorosítjátok, olvassátok át a feladatokat, ne maradjon közte olyan, amelynél bármi problémát találtok, illetve ne legyenek nagyon hasonló feladatok egy teszten belül, továbbá a csoportok nehézsége egyenlő legyen!

A feleletválasztós feladatoknak nagyobb a súlya, mert könnyebbek. Minden feladattípusnál levonás jár, ha nincs megoldva a feladat, vagy a megoldás hibás. Ha összpontszámra negatív érték adódna, akkor írhatunk helyette 0-t (remélhetőleg erre nem lesz szükség).

Az eredményekről részletes statisztikát kérek a mellékelt excel táblázat kitöltésével. Ez nagyon fontos, hogy a nagy ZH-ra/vizsgára ki tudjuk egyensúlyozni a teszt részt.

A következő kis ZH-t két hét múlva, a 8. héten írják a következő 3 tesztkérdés fejezetből (Adatbevitel billentyűzetről, kiírás képernyőre; A C nyelv utasításai; Függvények használata).

Röviden

Előadáson ezen a héten ment: típusok, pointerek, tömbök, dinamikus tömbök.

Pénteki csoportok feladata: állapotgép, többdimenziós tömbök. Házi feladat a múlt heti anyagban szereplő kapcsolódó feladatok, illetve az ugyancsak ott szereplő „A C nyelv utasításai” fejezet a tesztből.

Hétfői és szerdai csoportok feladata: Egyszerű függvények írása, pointerek és dinamikus tömbök. Házi feladat: az anyag végén látható feladatokból azt adjátok, amit jónak láttok. Továbbá a „Függvények használata” című fejezet a tesztkérdések közül.

1. ZH (használható hozzá kétoldalas C összefoglaló és kézzel írott puska).

Név:		FV sikeres:		× 3 pont=	
Neptun kód:		FV sikertelen:		×–3 pont=	
Hallgató aláírása:		MF sikeres:		× 2 pont=	
		MF sikertelen:		×–2 pont=	
		FV=FeleletVálasztós, MF=Megoldandó Feladat		Össz:	

Pontozás: -2: 1, 3-8: 2, 9-14: 3, 15-18: 4, 19-22: 5

Pénteki csoportok

A múlt heti gyakorlat anyaga az irányadó. A ZH-hoz tartozó tesztkérdések megbeszélése és a ZH megírása után beszéljük meg a házi feladatot, majd következzen az új anyag: állapotgép és többdimenziós tömbök. A kapcsolódó házi feladatok a múlt heti anyag végén találhatók. A teszt feladatok közül „A C nyelv utasításai” című fejezet következik.

Hétfői és szerdai csoportok

A ZH-hoz tartozó tesztkérdések megbeszélése és a ZH megírása után beszéljük meg a házi feladatot, majd következzen az új anyag:

Függvények használata

Eddig csak a main() függvény definíciójának megírásával találkoztak a hallgatók, ezentúl más függvényeket is fognak készíteni.

Írjunk függvényt, mely három egész szám közül kiválasztja a legnagyobbat! Hívjuk a main()-ből különféle paraméterezéssel, a visszatérési értéket különféleképp használjuk fel! Írjunk void visszatérési típusú függvényt: kiírja a sorszámot és az értéket. Hívjuk meg ezt is!

```
#include <stdio.h>

int legnagyobb(int a,int b,int c){
    if(a>b){
        if(a>c)return a;
        return c;
    }
    if(b>c)return b;
    return c;
}

void kiir(int sorszam,int ertek){
    printf("%d. %d\n",sorszam,ertek);
    return; // elhagyható
}

int main(){
    int a,b,c,d;
    printf("1. %d\n",legnagyobb(-533,57,1));
    a=9,b=-55,c=98;
    d=legnagyobb(a,b,c);
    printf("2. %d\n",d);
    d=legnagyobb(b,c,a);
    printf("3. %d\n",d);
    d=legnagyobb(c,100,a);
    printf("4. %d\n",d);
    kiir(5,legnagyobb(d,a,-3));
    return 0;
}
```

A függvény csak akkor végez beolvasást vagy kiírást, ha ez a feladata. ZH-n és vizsgán nagyon súlyos pontlevonás a következmény, ha felesleges I/O művelet történik.

Pointerek

Mutassunk példát pointer használatára!

a) Pointer egy változóra mutat, *p-vel elérhető a változó, meg is változtatható.
b) Pointer a tömb valahanyadik elemére mutat
c) Járjunk be egy tömböt indexszel és pointerrel, használjunk függvényt!

```
#include <stdio.h>

double minkeres1(double tomb[],int n){
// Át kell adni az elemszámot is! sizeof(tomb) egy pointer méretét adja!
    double min=tomb[0];
    int i; for(i=1;i<n;i++)if(tomb[i]<min)min=tomb[i];
    return min;
}

double minkeres2(double tomb[],int n){
    double *p,*pmin=tomb;//vagy &tomb[0]
    for(p=tomb+1;p-tomb<n;p++)if(*p<*pmin)pmin=p;
    return *pmin;
}

int main(){
    double x=99.5,*p, t[6]={3.14,-2.71,0,2008.1015,-612,1.34e-4};

    p=&x;   printf("*p=%g\n",*p);
    *p=-5;  printf("x=%g\n",x);
    p=t+5;  printf("*p=%g\n",*p);

    printf("1. %g\n",minkeres1(t,6));
    printf("2. %g\n",minkeres2(t,6));

    return 0;
}
```

Dinamikus tömbök

Írjunk programot, mely megkérdi egy poligon csúcsainak számát, majd létrehozza az x és y dinamikus tömböt a csúcsok koordinátáinak tárolására és fel is tölti a tömböt!

```
#include <stdio.h>
#include <stdlib.h>

int main(){
    int n,i;
    double * x, * y;
    printf("Hany csucsa legyen a poligonnak? ");  scanf("%d",&n);
    x=(double*)malloc(n*sizeof(double));
    y=(double*)malloc(n*sizeof(double));
    if(x==NULL||y==NULL){
        printf("Memoriafoglalasi hiba.");
        return -1;
    }
    for(i=0;i<n;i++){
        printf("%d. x=",i+1);  scanf("%lg",x+i);//vagy &x[i]
        printf("%d. y=",i+1);  scanf("%lg",y+i);
    }
    free(x);  free(y);
}
```

A dinamikusan lefoglalt memóriát sose felejtsék el felszabadítani!

Házi feladat

```
X ****************************************************************************
C Függvények használata
X ****************************************************************************
```

---

2. Mit ír ki a következő függvény, ha a bikini(14); utasítással hívjuk meg?

```
void bikini(unsigned n){while(n>>=1)putchar('X');}
```

---

3. Mit lesz az "a" változó értéke?

```
int g (unsigned i) { printf("%d\n", i); return i ?  g(i-1) : 0; }
//...
a = g(10);
```

[ ] 9, mert i = 10-el lett meghívva a függvény, és 10 nem egyenlő nullával, tehát 10 - 1 = 9 lesz a visszatérési érték.
[ ] Nem lehet megmondani, mert a függvény örökké saját magát hívogatja, amíg le nem állítják.
[ ] 0, mert a legkülső hívás ebben a rekurzióban mindig 0-val tér vissza.

---

4. Átalakítható-e egy ciklussal megadott programrészlet úgy, hogy abban a ciklust önhivatkozó (rekurzív) függvényhívás helyettesítse?

[ ] Igen.
[ ] Nem minden esetben.

---

5. Adott az alábbi függvénydeklaráció:

```
int compare(const void*, const void*);
```

Okoz-e fordítási idejű hibát az alábbi kódrészlet?

```
int a[] = {1,2,3};
int b[] = {4,5,6};
int r = compare(a, b);
```

[ ] Igen.
[ ] Nem.

---

6. Mit ír ki az alábbi kódrészlet?

```
int f(int param){
   printf("%d", param);
   return param+2;
}
...
x=f(5.7);
```

[ ] "5.7".
[ ] "5".
[ ] "6", mert 5,7 egészre kerekítve 6.

---

7. Mi a különbség a függvény és a makró között?

[ ] A függvény végrehajtása futtatáskor vezérlésátadással történik, míg a makró fordítás előtt helyettesítődik be.
[ ] A két elnevezés ugyanazt a fogalmat jelöli.

---

8. Mit ír ki a program?

```
main(int argc, char ** argv) { printf("%s",argv[2]); return 0; }
```

[ ] A program 2. paraméterét
[ ] A program 3. paraméterét
[ ] A program paramétereinek számát.

---

9. Mit ír ki a program?

```
main(int argc, char ** argv) { printf("%d\n", argc); return 0; }
```

[ ] A program 3. paraméterét
[ ] A program paramétereinek számát.
[ ] A program 2. paraméterét

---

10. Mit jelent az, ha egy globális változó statikus?

[ ] A változót kötelező inicializálni.
[ ] A változó nem érthető el más forrásállományokból.
[ ] A változó a program futása után is a memóriában marad és megőrzi értékét.

---

11. Mit jelent az, ha egy lokális változó statikus?

[ ] A változót kötelező inicializálni.
[ ] A változó a blokk befejeződése után is megőrzi értékét
[ ] A változó nem érthető el más forrásállományokból.

---

12. Milyen értékkel tér vissza f, ha f(25) paraméterekkel hívjuk meg?

```
int f(int a)
{
  int e = -1; int k = 1;
  while(a>=0) { a-=k; k+=2; e++; } return e;
}
```

---

13. Mit ír ki az alábbi program?

```
int main(){
  int x=1, y, z=(x=28, y=x+3);
  printf("%d", z);
}
```

---

14. Helyes-e az alábbi függvény és valóban a megkapott sztring hosszát írja-e ki?

```
int strlen( char s[] ){
   int i;
   for( i = 0; s[i] != 0; i++ );
   return i++;
}
```

[ ] Helyes a kód és a hosszt is jól számolja
[ ] Hibás a kód, mert nem adhatjuk vissza a lokálisan létrehozott változót.
[ ] A kód helyes, de nem adja vissza pontosan a sztring hosszát.

---

15. Mi lesz az fv(6) hívás eredménye?

```
int fv(int n) {
  if (n <= 1 ) return n;
  return fv(n - 1) + fv(n - 2);
}
```

---

16. Mit ír ki az alábbi program?

```
int cnt = 0;
int fv(int n) {
  cnt++; if (n <= 1) return n;
  return fv(n - 1) + fv(n - 2);
}
void main() { int a = fv(7);  printf("%d", cnt); }
```

---

17. s egy char[] típusú változó, amit a billentyűzetről olvastunk be.

```
Hogyan érdemes kiíratni?
[ ] printf("%s", s);
[ ] printf(s);
[ ] Mindkét megoldás egyformán jó.
-------------------------------------------------------------------------------
18. Helyes-e a következő program?
int valami(const int* i) { return *i = (*i)+1; }
int main(){ int i = 5; printf("%d", valami(&i)); return 0; }
[ ] Nem, futási idejű hiba
[ ] Igen
[ ] Nem, fordítási idejű hiba
-------------------------------------------------------------------------------
19. Ha egy függvény elé nem írunk visszatérési érték típust, akkor az
alapértelmezett típus:
[ ] int
[ ] void (nincs)
[ ] void *
-------------------------------------------------------------------------------
20. Milyen értéket ad vissza az alábbi függvény?

int increment(int a){ return a = a + 1; }
[ ] A függvény hibás, mert visszatérési érték helyett egy értékadás szerepel
a return után.
[ ] A paraméterben megadott értéknél eggyel nagyobbat.
[ ] A függvény hibás, mert a paraméter értéke nem változtatható meg.
-------------------------------------------------------------------------------
21. Mire használható a va_arg makró?
[ ] "Rendet rak" egy változó hosszúságú paraméterlistával rendelkező függvényből
való visszatérés előtt
[ ] Változó paraméterszámú függvényekben az argumentum pointer inicializálásához
[ ] Ezzel léphetünk tovább a következő azonosító nélküli argumentumra
-------------------------------------------------------------------------------
22. Problémát jelent-e, ha a programunk tartalmazza az alábbi két
függvénydeklarációt?
int sinus(int i);
int sinus(double d);
[ ] Igen, a C nyelvben a függvényneveknek egyedinek kell lenniük.
[ ] Igen, mert a szinusz függvény nem kaphat egészet bemenetként.
[ ] Nem, hiszen a paramétereik különbözősége miatt a fordító számára
egyértelműen
-------------------------------------------------------------------------------
23. Meghívható-e a main() függvény egy C programból?
[ ] Nem, mert nem tudjuk behelyettesíteni az argc, argv paramétereket.
[ ] Igen.
[ ] Nem, mert az rekurzióhoz vezetne, ahol fennáll a végtelen ciklus veszélye.
-------------------------------------------------------------------------------
24. Mit ír ki az alábbi függvény, ha meno("8GX61n6d") módon hívjuk meg?
void meno(const char * s){
    unsigned i;
    for(i=0;s[i];i++) {
      if(i%3)if(s[i]>='A'&&s[i]<='Z') s[i]=s[i]-'A'+'a';
      else if(s[i]>='a'&&s[i]<='z') s[i]=s[i]+'A'-'a';
    }
    printf("%s\n",s);
}

-------------------------------------------------------------------------------
25. Melyik main() deklaráció helyes az alábbiak közül?
A int main();
B int main(int argc);
C int main(int argc, char *argv[]);
[ ] Csak A és C
[ ] Mindhárom helyes
[ ] Csak A
-------------------------------------------------------------------------------
26. Melyik szabványos ANSI C függvény használható képernyőtörlésre?
[ ] Nincs szabványos C függvény a képernyőtörlésre
[ ] cls()
[ ] clrscr()
-------------------------------------------------------------------------------
27. Mit ír ki a program?
#include<stdio.h>
int main(int argc, char ** argv){
  if (!strcmp(argv[0],"fritillaria"))
    printf("Szojuz T-10\n");
  return 0;
}
[ ] Azt, hogy "Szojuz T-10".
[ ] Ha első paraméterként azt adtuk meg, hogy "fritillaria", akkor azt, hogy
"Szojuz T-10", különben semmit.
[ ] Ha a futtatható programállomány neve "fritillaria", akkor azt, hogy
"Szojuz T-10", különben semmit.
```

Példák

A dinamikus tömböket akkor is fel kell szabadítani, ha ezt a feladat nem írja, a tömb méretét akkor is át kell adni a függvénynek, ha ezt a feladat szövege nem írja,

1. Egészítsük ki a dinamikus tömbös feladatot egy olyan függvénnyel, mely kiszámítja a poligon kerületét!
2. Írjon függvényt, amely paraméterként kap két pozitív egész számot és visszaadja a legnagyobb közös osztójukat! Egészítse ki teljes programmá! A main() függvény kérjen be két számot, hívja ezzel paraméterezve a függvényt, ezután írja ki az eredményt!
3. Írjon függvényt, amely paraméterként kap egy double típusú elemeket tartalmazó tömböt és a tömb elemszámát, visszatérési értékként adja az elemek összegét! Egészítse ki teljes programmá! A main() függvény hozzon létre egy a felhasználó által megadott méretű dinamikus, double elemű tömböt, töltse fel a felhasználó által megadott adatokkal és hívja meg a függvényt. Az eredményt írja a szabványos kimenetre!
4. Írjon függvényt, amely paraméterként kap egy double típusú elemeket tartalmazó tömböt és a tömb elemszámát, visszatérési értékként logikai igaz vagy hamis értéket adjon annak megfelelően, hogy a tömb elemei monoton nőnek-e! Egészítse ki teljes programmá! A main() függvény hozzon létre egy a felhasználó által megadott méretű dinamikus, double elemű tömböt, töltse fel a felhasználó által megadott adatokkal és hívja meg a függvényt. Az eredményt írja a szabványos kimenetre!
5. Írjon függvényt, amely paraméterként kap egy egész elemeket tartalmazó tömböt és a tömb méretét (n), és feltölti az első n prímszámmal! Egészítse ki teljes programmá! A main() függvény hozzon létre egy a felhasználó által megadott méretű dinamikus, egész elemű tömböt, hívja meg a függvényt, és írjon ki a szabványos kimenetre csökkenő sorrendben minden második prímet a tömbből!