Programozás alapjai 1. (BMEVIEEA100)

Gyakorlat anyaga a 2. oktatási héten

Sok szeretettel üdvözlünk mindenkit az új félévben, jó munkát, kitartást, sok sikert!

Követelmények

A Programozás alapjai tárgy heti kétórás előadásból és heti kétórás gyakorlatból áll (szünet nélkül/szünettel: az oktató dönt). A félév során egy nagy zárthelyit írnak a Poppe tanár úr által ismertetett időpontban, továbbá a gyakorlatokon a TVSz-ben megengedett számú, minimum egy héttel a ZH megírása előtt bejelentett számú kis zárthelyi lesz. A minta nagy ZH csatolva.

A számonkéréseken használható a Szandi tanár úr által készített két oldalas C összefoglaló, valamint egy kézzel írott A/4-es lap (két oldalas).

Nagy ZH

A nagy ZH felépítése eltér a korábbi években alkalmazottól (félévismétlő hallgatók figyelmébe!). A zárthelyi beugróból és maximumrészből áll, akárcsak a vizsga. Aki a beugrót sikeresen teljesíti, annak sikeres a nagy zárthelyije, aki nem teljesíti a beugrót, annak sikertelen a nagy zárthelyije.

Mire jó a maximum rész? A beugró kispéldáira adott pontszámhoz hozzáadódik a maximum rész pontszáma, az így kapott pontszám alapján felállítjuk a hallgatók sorrendjét, ez alapján a sorrend alapján lehet majd menni elővizsgázni. Aki tehát nem akar elővizsgázni, annak felesleges maximumrészt írnia. Aki írt maximumrészt, de a beugrója sikertelen, annak nem javítjuk ki a maximumrészt. A maximumrész két összetettebb feladatból áll.

A minimumrész két részből áll, melyek sikeres volta logikai ÉS kapcsolatban áll egymással, azaz mindkét résznek külön-külön sikerülnie kell ahhoz, hogy valaki aláírást szerezzen.

A minimum rész első fele a teszt, melynek kérdései és válaszai publikusak. Szeptember végére készül el a teljes tesztfeladat-gyűjtemény, de addig is célszerű megadni a hallgatóknak az elérhetőséget, legegyszerűbb, ha minden oktató felteszi a honlapjára a csomag linkjét: http://www.eet.bme.hu/new/index.php?option=com_docman&task=doc_download&gid=97&Itemid=94. A link a tárgy hivatalos oldalán is megtalálható (http://www.eet.bme.hu/vieea100/index.html).

A tesztfeladatsor kétféle feladattípusból tevődik össze: 10 db feleletválasztós kérdésből, itt két vagy három lehetőség közül kell kiválasztani a jót. Jó válasz esetén +3 pont, rossz válasz vagy a válasz hiánya esetén -3 pont jár. Van további 5 db kódelemző feladat, ahol a kérdésre szöveges választ kell adni. Jó válasz esetén +2, rossz vagy hiányzó válasz esetén -2 pont jár.

A minimum rész második fele 3 db rövid program/programrészlet írása szöveges feladatmegadás alapján. A három kisfeladat a ZH anyagának első, második ill. harmadik egyharmadából kerül ki.

Kis ZH-k

A félév során a TVSz által megengedett maximális számú kis zárthelyit íratunk, melyek közül a TVSz-ben megadott számú legjobban sikerült zárthelyit vesszük figyelembe. A figyelembe vett zárthelyikre kapott osztályzatok átlaga legalább 2,0 kell legyen. A kis zh-k a

nagy zh-n és a vizsgán is használt tesztkérdésekből állnak, értékelésük kevésbé szigorú, mint a nagy zh-n ill. vizsgán, a pontos részletek az első kis zh megírása előtt kiderülnek.

Bejárás

A gyakorlatokon való részvétel kötelező, a TVSz által megadott mennyiségű (max. 30%) hiányzás megengedett.

Aláírás

Aláírást az kaphat, aki sikeres nagy zárthelyit ír, teljesíti a kis zh követelményt és rendszeresen bejár gyakorlatra.

Teremváltoztatás

Termet változtatni a fogadó oktató beleegyezésével lehet legkésőbb a harmadik oktatási hét végéig (szeptember 26.). A hallgatók a zh eredményüket a http://moodle.eet.bme.hu/ címen tudhatják majd meg, házi feladatot is itt tölthetnek majd fel, csoportba osztásuk szept. 26 után történik, utána már nem változtatunk (ugyanez igaz a Szoftver labor 1 tárgyra is). Az oktatók regisztrálásáról majd a későbbiekben beszélünk.

Házi feladatok

A hallgatók minden órán kapnak házi feladatot, ezek számonkérése olyan módon történjen, hogy a hallgatókat ösztönözze a feladat elkészítésére, de nem szükséges az oktatónak minden egyes megoldást ellenőriznie. A házi feladat az aktuális héthez tartozó tesztfeladatokat, valamint általában egy kisebb-nagyobb kidolgozandó programot/függvény tartalmaz, ezek megoldását az oktató által szükségesnek ítélt részletességgel a következő gyakorlaton meg kell beszélni.

Vizsga

A tárgy vizsgával illetve vizsgajeggyel zárul. A vizsgajegy az az osztályzat, amit a hallgató az írásbeli vizsgán megszerez, tehát nem számít bele sem a kis zh-k, sem a nagy zh osztályzata. A minta vizsga ugyancsak mellékelve van. Felépítése hasonlít a nagy zh-hoz, de bővebb tananyagot ölel fel. A vizsga beugróból és maximum részből áll.

A beugró részei a teszt és a kisfeladatok, akárcsak a zh-n. A teszt 5-5 feleletválasztós és szöveges választ igénylő feladatból áll, kisfeladatból pedig 3 van, akárcsak a nagy zh-ban. A maximum rész ezúttal három feladatot tartalmaz.

Pótlás

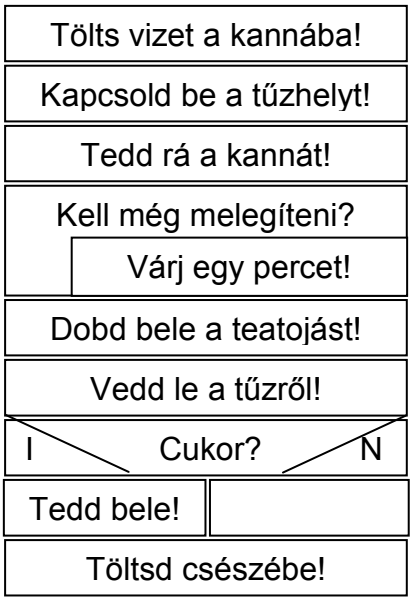
Kis zh-t pótolni nem lehet. Nagy ZH pótlására van lehetőség az utolsó előadás helyett megtartott pót zh és a vele együtt írt elővizsga alkalmával. Sikertelen vizsgát pótolni, vagy sikeres vizsgát javítani a TVSz előírásai szerint lehet.

1. gyakorlat

Mivel a gyakorlatot alapvetően tapasztalt oktatók tartják, az itt leírtaktól nyugodtan eltérhetnek, ha úgy tartják célszerűnek.

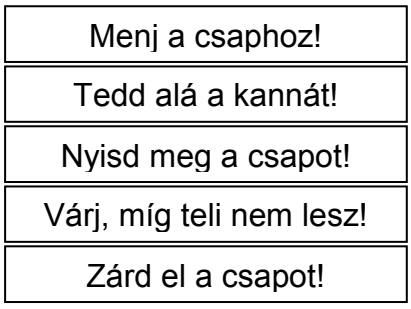
Előadáson nem jutott idő, ezért beszéljünk a Neumann elvről, a fordítás menetéről (C program => preprocesszor => fordító => linker). Az operációs rendszer szerepéről, az általa biztosított függvényekről. Írjuk fel a Hello world! példát és beszéljük meg a részleteit! Hívjuk fel a figyelmet, hogy bár a printf nagyon egyszerűnek tűnik, ahhoz, hogy a betűből képpont legyen, nagyon komoly háttér szükséges.

Beszéljünk a programtervezésről röviden. Mikor szükséges? Ha a megírni kívánt program mérete ezt megkívánja. Tervezéshez segítség a folyamatábra vagy a struktogram. Utóbbi merevebb, de könnyebb belőle strukturált kódot írni. Írjunk fel egy nem programozási példát, pl. teafőzés:



Ezen a példán sok mindent be lehet mutaatni:

- Utasítás, ciklus, feltételes elágazás
- Meg lehetett volna adni másképp is? Persze. Pl. filteres tea, elektromos teafőző, előbb vesszük le a tűzről, aztán megy bele a tojás stb. => egy problémát még pontos specifikáció esetén is számtalan módon meg lehet oldani, így könnyedén lebukhat az, aki a másikról másolja a feladatot a zh-n.
- Bonthatóak tovább is a blokkok? Persze. Például a víztöltés így is bontható:



- Meddig bontható? A végtelenségig. Meddig bontsuk? A rendelezésre álló utasításkészlet erejéig. Az áttekinthetőség érdekében, ha a diagramunk túl bonyolulttá válik, kiemelhetjük egyes részeit függvénnyé.

Osszunk ki egy összetettebb programrészletet, pl ami a következő oldalon látható!

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <ctype.h>

#define TESZTFAJL "teszt2008.txt"

char * magyarit(const char * src);

//*************************************************************************
void error(const char *s,...){
// Hibaüzenetet ír a stndard errorra és kilép.
// Hívás: mint a printf
//*************************************************************************
    va_list p;

    va_start(p,s);

    fprintf(stderr,"\n\nHiba: ");
    vfprintf(stderr,s,p);
    fprintf(stderr,"\n\n");

    va_end(p);
    getchar();
    printf(magyarit("Nyomjon <ENTER>-t a folytatáshoz!\n"));
    getchar();
    exit(-1);
}

//*************************************************************************
typedef enum{Csoport,Kerdes,Feladat,Hiba}ktip; // listaelem típusa
//*************************************************************************

//*************************************************************************
typedef struct lanc{       // kérdések láncolt listája
//*************************************************************************
    ktip tip;              // a kérdés típusa
    unsigned nv;           // válaszok száma kérdésnél, max 3
    char feladat[4000];    // [0]-> kérdés, feladat vagy a csoport neve
    char magy[1000];       // ide kerulnek a magyarazatok
    unsigned index;        // ha kérdés, akkor hanyadik a helyes válasz
    struct lanc * kov;     // a lista következő eleme
}kerdes,*pkerdes;

//*************************************************************************
void strmaxcat(char * dest,unsigned max,const char * src){
// Ha van elég hely, hozzáfűzi, ha nincs, hiba, betesz egy új sort az összefűzéshez!
//*************************************************************************
    size_t nd=strlen(dest);
    size_t ns=strlen(src);
    if(nd+ns+2>=max)error("Tul hosszu osszefuzes:\n%s\n+\n%s\n",dest,src);
    dest[nd]='\n';
    strcpy(dest+nd+1,src);
}

/*
//*************************************************************************
typedef struct{
//*************************************************************************
    unsigned sorrend;  //kérdésnél mi a képernyőre írás sorrendje
    unsigned jo;       // 0/1: rossz/jó
    unsigned szam;     //kérdésnél hányast választott, feladatnál mennyit írt be
    char szoveg[81];   //ha szöveges választ várunk,ezt adta meg
    unsigned beuns[5]; //feladatnál a behelyettesített számok
    char bechar[2][16];//feladatnál a behelyettesített stringek
    unsigned kiuns;    //feladatnál a helyes szám
    char kichar[40];   //feladatnál a helyes string
    unsigned eredmeny; // 0/1: szám/szöveg
}valasztipus;
*/

//*************************************************************************
int main(){
//*************************************************************************
    FILE * fp;
    char t[100];
    tesztfajl tf;
    unsigned i,j,n=1;
    typedef void (*fvt)(tesztfajl*);
    fvt tx[]={kerdescsop,zh,vizsga,zhkiir,vizsgakiir};
    srand((unsigned)time(NULL));

    // tesztfájl beolvasása

    fp=fopen(TESZTFAJL,"r");
    if(fp==NULL)error("main() -> Nem tudtam megnyitni a %s fajlt.",TESZTFAJL);
    beolvas(fp,&tf);
    fclose(fp);

    // Menü

    do{
        printf("****************************************"
               "****************************************\n");
        printf("%s\n",magyarit("Válasszon a szám beírásával"
               " és az <ENTER> leütésével!"));
        printf("%s\n",magyarit("  1 Kilépés"));
        printf("%s\n",magyarit("  2 Kérdéscsoport gyakorlása"));
        printf("%s\n",magyarit("  3 ZH feladatsor gyakorlása"));
        printf("%s\n",magyarit("  4 Vizsga feladatsor gyakorlása"));
        printf("%s\n",magyarit("  5 ZH feladatsor kiírása"));
        printf("%s\n",magyarit("  6 Vizsga feladatsor kiírása"));
        if(scanf("%s",t)!=1)n=1;
        else if(sscanf(t,"%u",&n)!=1)n=0;
        if(n>1&&n<7)tx[n-2](&tf);
    }while(n!=1);
/*
    for(i=0;i<tf.db;i++){
        printf("????????????????????????????????????????????????????\n");
        printf("%s\n",magyarit(tf.p[i]->feladat));
        if(tf.p[i]->magy[0])printf("%s\n",magyarit(tf.p[i]->magy));
    }
*/
    // felszabadítás

    for(i=0;i<tf.db;i++)free(tf.p[i]);
    free(tf.p);
    free(tf.pcsop);
    return 0;
}
```

Mutassuk meg a program részeit anélkül, hogy a működésről beszélnénk! (A fenti kód a teszt programból lett kihasítva.)

- #include szerepe, preprocesszor => nincs pontosvessző
- #define konstans definíció
- magyarit => függvénydeklaráció (prototípus) => csak a fejléc (visszatérési típus, név, paraméterek típusa esetleg neve is, utóbbi elhagyható).
- error => függvénydefiníció (ki van fejtve)
- megjegyzések fajtái és célja
  - /**/ => az eredeti C-ben csak ez van, több soros => csak kódrészletek eltávolítására használjuk (lásd a mintát), mert nem egymásba ágyazható (a buta preprocesszor kezeli), ha megjegyzéseket írnánk vele, akkor a hosszabb kódrészletek eltávolítása gondot okozna a */-ekbe való ütközés miatt.
  - // => C++-ból jött, a sor végéig tart, két célra használjuk: láthatóság/olvashatóság javításra és érthetőség javításra. Láthatóság javításra díszítésként, (//********** stb.), érthetőség javításra: magyarázó szövegek. Ugyanezt a célt szolgálják a beszédes változó/függvénynevek (azonosítók). Komoly programban az a, b, c stb. változónevek kerülendők, csak akkor használjuk, ha egyértelmű, mint jelent. (i, j, k ciklusban, x,y, z koordináták stb.)
- enum és struct adatszerkezet, előtte is és mögötte is lehet függvény
- függvényen belüli tagolás, behúzás (többsoros {} blokk esetén a belét beljebb kezdjük az átláthatóság érdekében. Helyes és helytelen alkalmazás lásd a tesztben.

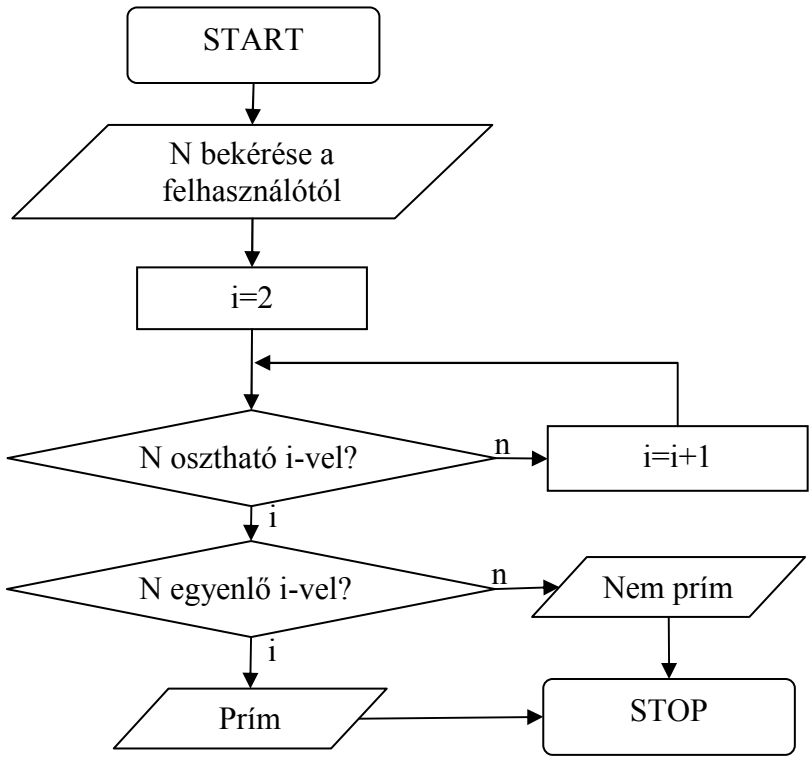
Írjunk programot, mely eldönti egy számról, hogy prím-e! Először kérdezzük meg a hallgatókat, hogy milyen módszert javasolnak! Valószínűleg felmerül az Eratoszthenészi szita ötlete, vagy ha nem, akkor mi is felvethetjük. Ehhez azonban tárolókapacitás szükséges. (Eratoszthenész szitája úgy működik, hogy felírjuk a számokat 2-től N-ig, aztán végigmegyünk a listán, és kihúzzuk 2 összes többszörösét. Ezután megkeressük az első ki nem húzott számot (ami egyébként prím), és ennek többszöröseit húzzuk ki. És így tovább. Ha eljutottunk N-ig, és még nincs kihúzva, akkor az prím.

Van-e egyszerűbb módszer? Igen. Menjünk el 2-től N-ig, és nézzük meg, hogy találtunk-e osztót. Ha a talált legkisebb osztó N, akkor prím, egyébként nem az. (Első körben tekintsünk el az optimalizációtól.)

Rajzoljuk fel a folyamatábrát vagy a struktogramot! (Lásd a következő oldalon.)

Írjuk fel a C kódot, és magyarázzuk el!

```c
#include <stdio.h>

int main(){
    unsigned i,N;

    printf("Kerek egy pozitiv egesz szamot! ");
    scanf("%u",&N);

    i=2;
    while(N%i!=0)i=i+1;

    if(i==N)printf("%u prim\n",N);
    else printf("%u nem prim, osztoja pl. %u\n",N,i);

    return 0;
}
```



Ha van még idő, gyorsítsunk a működésen, és

- menjünk N/2-ig
- menjünk sqrt(N-ig)

1.:

```
#include <stdio.h>

int main(){
    unsigned i,N;

    printf("Kerek egy pozitiv egesz szamot! ");
    scanf("%u",&N);

    i=2;
    while(N%i!=0 && i<=N/2)i=i+1;

    if(i>N/2)printf("%u prim\n",N);
    else printf("%u nem prim, osztoja pl. %u\n",N,i);

    return 0;
}
```

2.:

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

int main(){
    unsigned i,N,gy;

    printf("Kerek egy pozitiv egesz szamot! ");
    scanf("%u",&N);
```

```
    i=2;
    gy=sqrt(N); // egészre vág
//  gy=(unsigned)sqrt((double)N); // így nincs hibaüzenet
    while(N%i!=0 && i<=gy)i=i+1;

    if(i>gy)printf("%u prim\n",N);
    else printf("%u nem prim, osztoja pl. %u\n",N,i);

    return 0;
}
```

Házi feladat:

1. A teszt letöltése a http://www.eet.bme.hu/new/index.php?option=com_docman&task=doc_download&gid=97&Itemid=94 oldalról (vagy közvetett módon a http://www.eet.bme.hu/vieea100/index.html oldalról). Ezt lefordítani és futtatni (a `teszt2008.txt` legyen a projekt mappájában!) A „Kérdéscsoport gyakorlása” menüpontot választva a „Bevezetés” fejezet kérdéseit nézzék át és gyakorolják, a következő órán a gyakorlatvezetőkkel megbeszélik a megoldásokat.
2. Írjon programot C nyelven, mely kiszámítja két pozitív egész legnagyobb közös osztóját. Ehhez adhatunk folyamatábrát/struktogramot segítségként. A megoldást nézzük meg a jövő órán!

Javasolt a hallgatóktól kérni a megoldásokat, nevüket a névsorból olvasva, hogy memorizálhassuk a név-arc párosítást.

*